REMARKS

Please reconsider the application in view of the above amendments and the following remarks. Applicants thank the Examiner for carefully considering this application, for indicating that the drawings filed July 16, 2003 are accepted, and for acknowledging the IDS filed on July 22, 2004.

Disposition of Claims

Claims 1-23 are currently pending in this application. Claims 8-14 are canceled by this reply. Claims 24-28 are newly added by this reply. Claims 1, 15, and 24 are independent. The remaining claims depend, directly or indirectly, from claims 1, 15, and 24.

Claim Amendments

Claims 15-23 are amended to clarify the invention by reciting a computer-usable storage medium according to the Examiner's suggestion. Claims 24-28 are newly added. No new subject matter is added by the new claims as support for these claims may be found, for example, in at least Figure 3 and on pages 13-16 of the Specification.

Specification

The specification is objected to by the Examiner because U. S. Patent Application Serial Numbers are missing on page 1. The specification is amended by this reply to include the missing U. S. Patent Application Serial Numbers. Accordingly, withdrawal of this objection is respectfully requested.

Rejections under 35 U.S.C § 101

Claims 15-23 stand rejected under 35 U.S.C. § 101 because the language in the preamble recites "a computer-usable medium." Claims 15-23 are amended, in accordance with the Examiner's suggestion, to recite "a computer-usable storage medium." Accordingly, withdrawal of this rejection is respectfully requested.

Rejections under 35 U.S.C. § 102

Claims 1-23 are rejected under 35 U.S.C. 102(b) as being anticipated by "Programmer's Guide – iPlanet Portal Server: Mobile Access Pack," Sun Microsystems, Inc., published in November, 2001 ("iPlanet Portal Server"). Claims 8-14 are canceled by this reply, so the rejection is moot as to those claims. To the extent the rejection applies to the remaining claims, the rejection is respectfully traversed.

Turning to the rejection of the claims, for anticipation under 35 U.S.C. § 102, the reference must teach every aspect of the claimed invention either explicitly or impliedly. Any feature not directly taught must be inherently present. *See* MPEP § 2131. The Applicant respectfully asserts that iPlanet Portal Server does not disclose or suggest all of the limitations recited in the claimed invention.

Particularly, independent claims 1 and 15 recite, in part:

> *creating a node for said wireless client device in a software directory resident* on said server; and
> storing said information identifying properties of said wireless client device as attributes of said node in said software directory, wherein said information for said wireless client device is stored in *other than said XML form.*

(Emphasis added). iPlanet Portal Server does not disclose at least the limitations "creating a node for said wireless device in a software directory" or storing properties of the wireless device as attributes of the node "in other than XML form."

Applicants assert the Examiner is erroneously equating the Client Data object disclosed by iPlanet Portal Server to a node in a software directory. *See* Office Action dated November 1, 2006 at page 3. iPlanet Portal server merely teaches that characteristics of a mobile device are stored in a Client Data object. *See* iPlanet Portal Server at page 16. iPlanet Portal Server is completely silent regarding this object being a "node in a software directory" as required by claims 1 and 15. By interpreting a bare teaching of a Client Data *object* as disclosing *a node in a software directory*, the Examiner is either mischaracterizing the prior art or reading the claim language overly broad, both of which are wholly improper.

Moreover, even assuming *arguendo* that a Client Data object as disclosed by iPlanet Portal Server is a node in a software directory, iPlanet Portal Server does not teach that an attribute of a Client Data object is stored in any format other than XML format, as asserted by the Examiner. The portion of iPlanet Portal Server relied on by the Examiner merely discloses that a value of an attribute as retrieved from a Client Data object using the disclosed Java API is of type string. *See e.g.*, iPlanet Portal Server at page 290, the definition of the getProperty method. The fact that a value returned by an API method is of type string discloses nothing about the content of the string and nothing about how the information that yielded the value is actually stored in the Client Data object.

Further, iPlanet Portal Server clearly discloses that attributes of client devices are only stored in XML form. In the same paragraph with the bare disclosure of a Client Data object, iPlanet Portal Server teaches that detailed information on client data is found in "Appendix A 'API Descriptions,' Appendix B 'Sample Tasks,' and Chapter 8 'Attributes and Schemas.'" *See e.g.*, iPlanet Portal Server at page 16. Chapter 8 of iPlanet Portal Server, a copy of which is attached for the convenience of the Examiner, shows the attribute definitions for each of the client devices supported by a portal server. Each of these attributes is very clearly only in XML format. In addition, Appendix B of iPlanet Portal Server, a copy of which is also attached for the convenience of the Examiner, provides further evidence that the attributes are stored only in XML format. The sections on modifying and adding attributes very clearly refer to accessing only XML files to make the modifications and additions. *See* iPlanet Portal Server at pages 257-258.

In view of the above, iPlanet Portal Server clearly fails to disclose all the limitations recited in independent claims 1 and 15. Dependent claims 2-7 and 16-23 are patentable for at least the same reasons. Accordingly, withdrawal of this rejection is respectfully requested.

New Claims

Newly added independent claim 24 recites similar subject matter as independent claims 1 and 15 (*i.e.,* that the information is stored in other than XML form) and is patentable over iPlanet Portal Server for at least the same reasons described above. In addition, newly added independent claim 24 recites, in part:

> an *LDAP subschema* configured to: create a node in the LDAP software directory for said wireless client device and store said properties as attributes of the node; and create a Directory Information Tree (DIT) for the wireless client device.

(Emphasis added). iPlanet Portal Server fails to disclose or suggest an LDAP subschema and a DIT as recited in claim 24.

The Examiner cites page 21, lines 1-12 of iPlanet Portal Server as disclosing an LDAP directory and a DIT. *See* Office Action mailed November 1, 2006 on page 4. The cited portion of iPlanet Portal Server merely discloses that authentication of a client can take place using an LDAP pluggable authentication module. iPlanet Portal Server is completely silent with respect to an LDAP subschema that makes each client device a node in an LDAP software directory. Further, iPlanet Portal Server fails to disclose or suggest that LDAP subschema are used to create a DIT. In fact, iPlanet Portal Server does not even mention a DIT. The mere recitation of the phrase "LDAP" does not indicate that the LDAP authentication module performs any of the functionalities specified in newly added independent claim 24.

In view of the above, favorable consideration of newly added claim 24 is respectfully requested. Dependent claims 25-28, which depend from new independent claim 24, are patentable for at least the same reasons.

Conclusion

Applicants believe this reply is fully responsive to all outstanding issues and places this application in condition for allowance. If this belief is incorrect, or other issues arise, the Examiner is encouraged to contact the undersigned or his associates at the telephone number listed below.

Please apply any charges not covered, or any credits, to Deposit Account 50-0591 (Reference Number 03226/512001).

Dated: February 1, 2007

Respectfully submitted,

By [illegible] #50272

for Robert P. Lord
Registration No.: 46,479
OSHA · LIANG LLP
1221 McKinney St., Suite 2800
Houston, Texas 77010
(713) 228-8600
(713) 228-8778 (Fax)
Attorney for Applicants

Attachments (Chapter 8 of iPlanet Portal Server)
(Appendix B of iPlanet Portal Server)

196243_1

Appendix B

Sample Tasks

The various sections in this appendix include sample tasks for:

- Configuring the Attributes
- Customizing the Templates
- Adding a New ClientType
- Customizing the Client Detection Interface
- Creating Help Links
- Making a Provider Client Aware
- Implementing the Client Data API

Use these sample configuration scenarios to set up and customize your Mobile Access Pack server. Note that many of the tasks discussed in this chapter can be performed from the administration console and for more information on performing these tasks from the administration console, please refer to the Mobile Access Pack Administration Guide.

Configuring the Attributes

This section discusses how to modify existing component attributes and how to add new attributes to a component.

Modifying the Attributes

This section discusses how to modify attributes in the component XML file. To modify the attributes:

1. Log in to the machine running the Mobile Access Pack server and become superuser.

2. Enter:

```
# <iPS_install_base>/SUNWips/bin/ipsadmin get component
<component_name> > <filename>
```

   Here, replace:

   - *iPS_install_base* with the installed location of the iPlanet Portal Server
   - *component_name* with the name of the profile component to customize. For example, replace *component_name* with `iwtMailProvider` to customize the mail channel.
   - *filename* with the name of the file to redirect the output to

3. Open the file and edit the attributes in the file containing the component attributes.

   This is the file to which the `ipsadmin` command output was directed to.

4. Save and close the file.

5. Enter:

```
# <iPS_install_base>/SUNWips/bin/ipsadmin change component
<component_name> <filename>
```

   Here, replace:

   - *iPS_install_base* with the installed location of the iPlanet Portal Server.
   - *component_name* with the name of the component that was customized.
   - *filename* with the name of the file that contains the modified attribute values.

6. Restart the server.

Adding an Attribute

To add an attribute to a Mobile Access Pack server component:

1. Log in to the machine running the Mobile Access Pack server and become superuser.

2. Enter:

```
# <iPS_install_base>/SUNWips/bin/ipsadmin delete component
<component_name>
```

   Here, replace:

- *iPS_install_base* with the installed location of the iPlanet Portal Server
- *component_name* with the name of the profile component to update. For example, replace *component_name* with `iwtClient`.

This removes the current portal profile for the component.

3. Open the component XML file in `/etc/opt/SUNWips/xml` directory and add the new attribute to this file.

   For backup purposes, make a copy of the original XML file before adding the new attribute. For example, enter:

   ```
   # cp /etc/opt/SUNWips/xml/iwtClient.xml OldiwtClient.xml
   ```

4. Import the newly updated component XML file in to the portal profile settings. For example, to import, enter:

   ```
   # <iPS_install_base>/SUNWips/bin/ipsadmin -import
   <component_name>.xml
   ```

   Here, replace:

   - *iPS_install_base* with the installed location of the iPlanet Portal Server
   - *component_name* with the name of the profile component. For example, replace *component_name* with `iwtClient`.

5. Restart the server.

Customizing the Templates

When modifying template files, please follow the following recommendations:

1. Log in to the Mobile Access Pack server and become superuser.
2. Create a directory similar to the `default` directory in `/etc/opt/SUNWips/desktop/default`. For example:

   a. Change directories to `/etc/opt/SUNWips/desktop/default`.

   b. Enter:

   ```
   # mkdir NewDesktop
   ```

3. Copy files from the `default` directory in `/etc/opt/SUNWips/desktop/default` over to the new directory. For example, enter:

```
# cp -R default/* NewDesktop
```

Although you can copy all the files under the `default` directory, we recommend you only copy files you wish to change and/or customize over to the `NewDesktop` directory. When copying files, use the same directory hierarchy to the template files.

4. Modify the copied files in the new directory.

   Do not modify files under the `default` directory.

5. Modify the `iwtDesktop-type` attribute. That is, enter the path to the directory that contains the customized desktop templates in this attribute. For example, specify the path to the `NewDesktop`.

   The path must be relative to `/etc/opt/SUNWips/desktop`. If the server cannot find the required files under the directory you specify here, it will default to the files in the `default` directory.

6. Restart the server.

NOTE	Do not modify the original template files in the `default` directory as this will break uninstall and/or upgrade.

Adding a New ClientType

The Mobile Access Pack server includes support for generic HTML, cHTML, and WML clients. In order to extend support for other markup languages (or clientTypes), for example, XYZ markup language, refer to the sample XYZ configuration scenario in this section.

For information on adding a clientType from the administration console, please see Chapter 3 in the Mobile Access Pack Administration Guide. From the command line:

1. Log in to the Mobile Access Pack server machine and become super user.
2. Change directories to `/etc/opt/SUNWips`.

   Develop the appropriate templates to translate the content produced by the Mobile Access Pack server providers for display on an xyz client.

Configuring Authentication for XYZ Clients:

Assuming that XYZ clients are not a derivative of WML or cHTML:

NOTE If the XYZ client is a derivative of cHTML or WML, create the xyz directory under WML or cHTML (accordingly) and copy only the files you wish to customize for the XYZ clientType into the xyz directory.

a. Change directories to `auth/default` and create a directory for XYZ. That is, enter:

```
# cd auth/default
# mkdir xyz
```

b. Copy files from either `html`, `wml`, or `chtml` directory to the `xyz` directory. That is, enter:

```
# cd xyz
# cp -R ../chtml/* .
```

Do not modify or change the name of the files copied from the chtml directory.

c. Modify and save the `xyz` template files to support XYZ clients.

See "Authentication Template Files," on page 21 for more information on template files in this directory.

d. Specify the supported authentication modules for the XYZ clients. That is:

From the Administration Console:

I. Log in to the administration console and select Manage Platform Settings.

II. Select Authentication and Show Advanced Settings.

III. Specify the supported authentication modules for XYZ clients under Supported Auth Modules for Clients.

For example, enter `xyz | Ldap` and select Add.

IV. Select Submit.

From the command line, incorporate the supported authentication modules for XYZ clients in the `iwtAuth-supportedAuthModules` attribute using the `ipsadmin` command.

See Chapter 8, "Attributes and Schemas" for more information on modifying this attribute.

e. Specify the `loginWorkerClass` for XYZ clients. That is, modify the `iwtAuth-loginWorkerClasses` attribute value to include the newly added clientType. See `iwtAuth-loginWorkerClasses` for more information.

   See Chapter 8, "Attributes and Schemas" for more information on this attribute.

Configuring Channels for XYZ clients:
Do not modify files in the `/etc/opt/SUNWips/desktop/<default>` directory. See "Modifying the Attributes," on page 257 for more information.

a. Change directories to `desktop/<NewDesktop>/`. That is, enter:

```
# cd /etc/opt/SUNWips/<NewDesktop>
```

b. Create a XYZ directory under the provider directory.

   For each provider content that will be delivered to the XYZ client, create a subdirectory for the XYZ client under that provider directory. For example, to provide mail channel on the XYZ client:

   I. Change directories to the mail provider directory. Enter:

```
# cd iwtMailProvider
```

   II. Create a directory for XYZ clients here. Enter:

```
# mkdir xyz
```

   III. Copy files from html, chtml, or wml directory over to the xyz directory. For example, enter:

```
# cd xyz
# cp -R ../chtml/* .
```

   IV. Modify and save the template files to support clients that use the XYZ markup language.

   Refer to Chapter 5, "Application JSPs" and Chapter 4, "Content Provider Template Files" for more information on these template files.

3. Modify the `iwtClient-clientTypes` attribute to define device characteristics of XYZ clients.

   See "Client Detection Module Attributes," on page 237 and modify this attribute accordingly. For example, to modify this attribute, enter:

```
<VAL>clientType=XYZ|userAgent=XYZ|contentType=text/xyz|fileIdentifi
er=xyz|filePath=my_xyz|charset=ISO-8859-1;UTF-8</VAL>
```

This adds the XYZ clientType and includes important client specific information to support the XYZ clients.

Enabling Content Delivery for XYZ Clients:

In order to set up and deliver content via channels to the newly configured XYZ client:

1. Log in to the Mobile Access Pack server machine and become super user.
2. Modify the `iwtDesktop-clientAllProviders`, `iwtDesktop-clientUserSelectedProviders`, and `iwtDesktop-clientChannelListModules` attributes. To modify, see "Configuring the Attributes," on page 257 for more information. See also "Desktop Attributes," on page 241 for information on:
    - `iwtDesktop-clientAllProviders` and enter:

      `xyz|iwtUserInfoProvider;iwtXMLProvider;iwtPostitProvider`

      This indicates that a user information channel, an XML channel, and a PostIt channel are supported for a xyz client.
    - `iwtDesktop-clientUserSelectedProviders` and enter:

      `xyz|iwtUserInfoProvider`

      This makes the xyz client available to the end user from the Mobile Devices page on their desktop.
    - `iwtDesktop-clientChannelListModules` and enter:

```
xyz|com.iplanet.portalserver.desktop.util.channellist.ClientChannel
List.
```

This adds the clientType to the Client Specific Channel List Module.

The xyz client is now displayed in the Mobile Devices page from the user's desktop page and it has access to all of the channels that were made available to it.

Customizing the Client Detection Interface

This section provides instructions for:

- Developing a Customized Client Detector

- Modifying the Default Client Detector

Developing a Customized Client Detector

Client detection can be achieved by creating a client detector object that will:

1. Parse data from the HTTPServletRequest of the requesting client
2. Perform some method of matching based on the stored clientTypes and their corresponding properties

In order to develop such a client detector, the client detection class must:

- Implement an interface called ClientDetectionInterface.
- Contain a method called `getClientType` which will be passed to the HTTPServletRequest from the requesting client.

The class can then look at the header information provided by the client in it's request, match it against the clientType and its device specific characteristics saved in the Client Data, and return the clientType that it has received from the Client Data object.

Sample Client Detection Interface

```
public class SampleCDM implements ClientDetectionInterface {

/*

* Detects the client type based on the Request

* @param request HttpServletRequest

* @return a String representing the clientType

* @exception ClientDetectionException if there is an error
retrieving the client data.

*/

   public String getClientType(HttpServletRequest request) throws

   ClientDetectionException{

   }

   public String HTTPheaderCheck(String HTTPheader){

   }

}
```

See Appendix A, "API Descriptions" for detailed information on the Mobile Access Pack server client detection interface. After developing the client detector class:

1. Change the profile settings to enable client detection using the newly developed class. To change:
   a. Modify the `iwtAuth-clientDetectionEnabled` attribute value.

      See "iwtAuth-clientDetectionEnabled," on page 234 and modify this attribute value by referring to "Configuring the Attributes," on page 257.
   b. Modify the `iwtAuth-clientDetectionClass` attribute to use the newly developed client detector class.

      See "iwtAuth-clientDetectionClass," on page 234 for more information on this attribute and modify this attribute value by referring to "Configuring the Attributes," on page 257.
2. Ensure that the class is placed in the webserver's classpath.
3. Restart the server.

Modifying the Default Client Detector

By default, the Mobile Access Pack client detection module performs a substring matching to match the requesting client's HTTP header value (or userAgent) against the Client Data objects. To change the module to use regular expression matching:

1. Write a class that makes use of the ClientDetectionInterface.

```
public class SampleCDM implements ClientDetectionInterface {
    public String getClientType (HttpServletRequest req) throws
    ClientDetectionException{
        // get the user-agent from the HttpServletRequest
        String clientType = userAgentCheck(String user-agent);
        if ((clientType.equals("")) || (clientType == null))
            return
            Client.getDefaultInstance().getProperty("clientType");
        else
            return clientType;
        }
    public String userAgentCheck(String userAgent){
        // Implement the regular expression checking within this
        // method
        // return the clientType if it matches, empty String if it
        // does not match
    }
}
```

2. Ensure that the class is placed in the webserver's classpath
3. Login to the Administration Console and select Manage Platform Settings.
4. Select Authentication and select Show Advanced Settings.
5. Ensure that Client Detection is enabled.
6. Add the customized Client Detection class into the Client Detector class attribute.
7. Restart the server.

Creating Help Links

In order to provide assistance with content to the mobile device users, the Mobile Access Pack server provides a mechanism to display help for the user's home page and for each channel in the appropriate markup language.

Enabling Help Content Delivery on a Mobile Device

This section discusses how to deliver help content to a mobile device user.

Each provider uses the Provider API (PAPI) interface to find which methods are called by the various desktop classes to create the desktop. The two methods, `hasHelp()` and `getHelp()`, are used by the Mobile Access Pack server to display a help link for each provider. See "Client Help," on page 251 for more information on these methods.

When configuring the help link for a channel:

1. Create the `hasHelp()` and `getHelp()` methods for each supported provider. Use the sample methods below when writing these methods.

   The Mobile Access Pack server providers include these methods in the class file. Develop only the help content for the Mobile Access Pack server providers and proceed to Step 2. The sample methods discussed here are only for providers who do not have the help mechanism built into them.

Sample hasHelp() method

The `hasHelp()` method must return the boolean value of true or false depending on whether or not the attribute `iwtComponent-clientHelpLinks` is found in the profile for the component.

```
public class WirelessSampleProvider() extends SampleProvider{
    public boolean hasHelp(){
        Vector clientLinks = getListProperty("iwtSampleProvider",
            "clientHelpLinks", null);
        if (clientHelpLinks == null){
            return false;
        }
        for (Enumeration e = clientHelpLinks.elements();
            e.hasMoreElements();){
            if ((String)e.nextElement() == clientType){
                return true;
            }
        }
        return false;
    }
}
```

Sample getHelp() method

The `getHelp()` method must return the URL that corresponds to the specified clientType.

```
public class SampleProvider() extends ProfileProviderAdapter{
    public boolean getHelp(){
        Vector clientLinks = getListProperty("iwtSampleProvider",
            "clientHelpLinks", null);
        if (clientHelpLinks == null){
            return false;
        }
        for (Enumeration e = clientHelpLinks.elements();
            e.hasMoreElements();){
```

```
            String link = (String)e.nextElement();
            String url = "";
            if (link.indexOf(clientType) > -1){
                StringTokenizer tok = new StringTokenizer(link, "|");
                String temp = tok.nextToken();
                    if (temp.equals(clientType)){
                        url = tok.nextToken();
                    }
            }
        }
        String userLocale = getStringProperty("iwtUser", "locale",
            "en_US");
        String localizedUrl = "/docs/" + userLocale+ "/online_help/"
            + url;
        String proto = SystemProperties.get("ips.server.protocol");
        String host = SystemProperties.get("ips.server.hostname");
        String port = SystemProperties.get("ips.server.port");
        URL help = null;
        try{
            help = new URL(proto, host, Integer.parseInt(port),
                localizedUrl);
        } catch (MalformedURLException mue){
            debug.error("Unable to create help link");
            help =  null;
        }
        return help;
    }
}
```

2. Add the `clientHelpLinks` attribute (see `<channelName>-clientHelpLinks` for more information) to the profile for the component. To add, refer to "Configuring the Attributes," on page 257 for more information.

3. Restart the server.

   Help is now made available in the user's mobile device home page under the Options selection.

Developing and Storing the Help Files

Develop help content for all supported clientTypes and their channels and store the files under `<iPS_install_base>/SUNWips/public_html/docs/<locale>/online_help`.

Making a Provider Client Aware

This section discusses how to make a provider serve content to multiple clientTypes (such as cHTML and WML). In order to service multiple clients, a provider must become aware of the type of client requesting its service; that is, a provider must become client aware. Use the sample set up scenario in this section when making a provider client aware.

The sample HelloWorld3 Provider (`HelloWorld3Provider`) in (`<iPS_install_base>/SUNWips/sample/desktop/classes/com/iplanet/portalserver/providers/helloworld3`) is the provider used in our example here.

In this example:

Content Delivery

The `WirelessHelloWorld3Provider` uses template files to display its content and to determine its presentability. That is, if template for a specific clientType is available, the `WirelessHelloWorld3Provider` will support display of content on that clientType.

Package

The `WirelessHelloWorld3Provider` resides in the same package as the other iPlanet Portal Server providers:

```
package com.iplanet.portalserver.providers.helloworld3;
```

Inheritance

Inheritance provides the provider with the functionality of the Provider API (PAPI). The `WirelessHelloWorld3Provider` inherits from the existing `HelloWorld3Provider` provider.

```
public class WirelessHelloWorld3Provider extends HelloWorld3Provider
```

Class Variables

The `WirelessHelloWorld3Provider` contains the following class variables and includes a variable for the one generic content template that is used in the `getContent()` method.

```
private static final String contentTemplate = "display.template";

private String  clientType = null;

private boolean isGenericHTML;

private boolean isPresentable;
```

Init Method

```
public void init(String n, Session s) throws ProviderException
```

ClientType

ClientType refers to the list of clients (such as HTML, cHTML, and/or WML) supported by the provider. Retrieval of the clientType value stored in the session is necessary to determine the value of the `isGenericHTML` attribute. This attribute is used to determine whether or not to invoke the superclass method.

```
clientType    = "";

isGenericHTML = false;

try {

   clientType    = s.getProperty("ips.clientType");

   isGenericHTML = (Client.getInstance(clientType).

      getProperty("genericHTML")).equals("true");

} catch (SessionException se) {

   throw new ProviderException("Unable to retrieve Session client

      type");

} catch (ClientException ce) {}
```

Presentability

Presentability refers to determining whether or not to service the client based on whether or not the requesting clientType is supported by the provider. That is, the `WirelessHelloWorld3Provider` is presentable if a template is found for the requesting clientType.

Presentability, in this example, is based on the following logic:

```
* If the content template is found, then the channel is presentable
to the client.
```

```
Or,

* If isGenericHTML is true, then the superclass method will be used,
and the client is assumed to be presentable.

isPresentable = true;

StringBuffer template = null;

if (!isGenericHTML) {

    try {

        template = getTemplate(contentTemplate);

    } catch (ProviderException e) {

        isPresentable = false;

    }

    if (template == null) {

        isPresentable = false;

    }

}
```

Content Retrieval

Content retrieval is based on the `getContent()` method.

```
public StringBuffer getContent(Map m) throws ProviderException
```

First, check to see if the client is a generic HTML client, defined by the `isGenericHTML` attribute. If the boolean value is true, then invoke the superclass `getContent()` method.

```
if (isGenericHTML)

    return super.getContent(m);
```

If the value of the `isGenericHTML` variable is false, then retrieve the client specific content template. The `getTemplate()` method will use the client's filePath attribute in the Client Data to locate the template.

```
StringBuffer content  = getTemplate(contentTemplate);

return content;
```

The following will determine if the provider is presentable:

```
public boolean isPresentable() {

    return isPresentable;

}
```

The following method, required as defined by the Provider interface, provides the provider with a mechanism to indicate whether or not it is editable from a particular clientType. If this method returns true, then there must be a client aware `getEdit()` method.

```
/*
* Determine if the provider is editable
* @return boolean
*/
public boolean isEditable() {
    if (isGenericHTML)
    return super.isEditable();
    return false;
}
```

Installation

It is assumed that the existing `iwtHelloWorld3Provider` component has been installed on the system.

Sample iwtHelloWorld3Provider Template Files

Create the client specific template files for this provider. The template files are typically located in `/etc/opt/SUNWips/desktop/default`. In order to set up the template files, enter:

```
# mkdir iwtHelloWorld3Provider
# mkdir /iwtHelloWorld3Provider/wml
# touch /iwtHelloWorld3Provider/wml/display.template
# mkdir /iwtHelloWorld3Provider/chtml
# touch /iwtHelloWorld3Provider/chtml/display.template
```

The `display.template` file contains something similar to the following:

```
# cat ./iwtHelloWorld3Provider/wml/display.template
<p>
Wireless Hello World #3!
</p>
# cat ./iwtHelloWorld3Provider/chtml/display.template
Wireless Hello World #3!
```

```
<br>
```

Setup

1. Log in to the Mobile Access Pack server administration console and select Manage Domains.
2. Select the domain and expand Applications.
3. Select Desktop and select `iwtHelloWorld3Provider` from the list of Available Channels.
4. Select the Edit Channel button to display the profile settings for the provider.
5. Change the Provider Class Name to `com.iplanet.portalserver.providers.helloworld3.WirelessHelloWorld3Provider` and select Submit.

   This will enable use of the new class that provides the client awareness.
6. Select Show Advanced Options and modify the Client Specific Available Providers for the clients supported by this provider. For Example,

   `WML|iwtHelloWorld3Provider`

   `cHTML|iwtHelloWorld3Provider`
7. Select Submit.

Sample WirelessHelloWorld3Provider Class File

Create the `WirelessHelloWorld3Provider` class file or modify a copy of the `HelloWorld3Provider` class file in `iPS_install_base/SUNWips/sample/desktop/classes/com/iplanet/portalserver/providers/helloworld3` directory.

```
package com.iplanet.portalserver.providers.helloworld3;

import java.util.*;

import com.iplanet.portalserver.client.Client;

import com.iplanet.portalserver.client.ClientException;

import
com.iplanet.portalserver.providers.helloworld3.HelloWorld3Provider;

import com.iplanet.portalserver.providers.ProfileProviderAdapter;

import com.iplanet.portalserver.providers.ProviderException;

import com.iplanet.portalserver.session.Session;

import com.iplanet.portalserver.session.SessionID;
import com.iplanet.portalserver.session.SessionException;
/*
```

The following example shows how to use the existing `HelloWorld3Provider` to create a new Mobile Access Pack provider that is client aware.

```
*/
public class WirelessHelloWorld3Provider extends
HelloWorld3Provider{
    private static final String contentTemplate = "display.template";
    private String  clientType = null;
    private boolean isGenericHTML;
    private boolean isPresentable;
/*
* Constructor
*/
    public WirelessHelloWorld3Provider() {}
/*
* Initialize the provider
* Presentability of the provider to the desktop is based on the
client type and content template retrieval
* @param n Component name
* @param s Session
* @exception ProviderException
*/
    public void init(String n, Session s) throws ProviderException {
        super.init(n, s);
        clientType    = "";
        isGenericHTML = false;
        isPresentable = true;
        try {
            clientType    = s.getProperty("ips.clientType");
            isGenericHTML = (Client.getInstance(clientType).
                getProperty("genericHTML")).equals("true");
        } catch (SessionException se) {
            throw new ProviderException("Unable to retrieve Session
                client type");
        } catch (ClientException ce) {}
/*
* Determine presentability.
* if (content template found), then the channel is presentable
* if genericHTML is true, then use parent methods for html markup
and do not check for the template.
*/
    StringBuffer template = null;
    if (!isGenericHTML) {
        try {
            template = getTemplate(contentTemplate);
        } catch (ProviderException e) {
            isPresentable = false;
        }
        if (template == null) {
            isPresentable = false;
        }
    }
/*
* Get content for the client.
* @param  m
* @return StringBuffer of the provider content.
* @exception ProviderException
*/
    public StringBuffer getContent(Map m) throws ProviderException {
        if (isGenericHTML) {
            return super.getContent(m);
        }
        StringBuffer content  = getTemplate(contentTemplate);
        return content;
    }
/*
* Determine if the provider is editable
* @return boolean
*/
    public boolean isEditable() {
        if (isGenericHTML) {
            return super.isEditable();
        }
        return false;
    }
/*
* Determine if the provider is presentable
* @return boolean
*/
    public boolean isPresentable() {
        return isPresentable;
    }
}
```

Implementing the Client Data API

In order to access the Client Data and retrieve any Client Data for the device accessing the desktop, within the Provider, develop a method to:

1. Retrieve the clientType string value from the Session using the `ips.clientType` property.
2. Retrieve the client instance using the clientType value retrieved from the Session.

The client instance can be used to access the client's properties in the Client Data via the Client API. For more information, see "Client Data," on page 245.

Initializing a Provider with the Init Method

The following is an example usage of the Client API that relies on the init method defined in the Provider API (PAPI).

```
com.iplanet.portalserver.providers
```

Class ProfileProviderAdapter

```
java.lang.Object
  |
  +--com.iplanet.portalserver.providers.ProviderAdapter
        |
        +--com.iplanet.portalserver.providers.ProfileProviderAdapter
```

init

```
public void init(java.lang.String n, Session s)
          throws ProviderException
```

This initializes the provider. If additional initialization is required by the provider, this method can be overriden. However, the overriding `init()` call must always have `super.init()` as the first statement executed.

Parameters:

```
n - The unique indentifying name for this provider.
s - The user's session.
```

Example:

A following is an example implementation of this method in a non-wireless provider:

```
public void init(String n, Session s) throws ProviderException {
    super.init(n, s);
}
```

Using the Session API to Retrieve the Client Type

In order to make a provider client aware, retrieve the clientType from the Session. This clientType represents the type of client accessing the desktop. The clientType value is used as an index to retrieve appropriate Client Data objects which is subsequently used to access the client's file path and device specfic template files using the file lookup mechanism.

```
com.iplanet.portalserver.session
```

Class Session

```
java.lang.Object
  |
  +--com.iplanet.portalserver.session.Session
```

Use the `getProperty()` method to retrieve the clientType. Use the `ips.clientType` property name and retrieve the clientType value stored in the Session. The following example retrieves the clientType value within the Provider's `init()` method.

```
/*
* Gets the property stored in this session.
* @param name The property name.
* @return The property value in String format.
* @exception A SessionException is thrown if the session reached its
maximum
* session time, or the session was destroyed, or there was an error
during
* communication with session service.
*/
public String getProperty(String name) throws SessionException
```

Example:

```
private String clientType = null;
public void init(String n, Session s) throws ProviderException {
    super.init(n, s);
    clientType = "";
    try {
        clientType = s.getProperty("ips.clientType");
    } catch (SessionException se) {
        throw new ProviderException("Unable to retrieve Session
            client type");
    }
}
```

The provider can also be made to use the inherited PAPI `getSession()` method to retrieve the Session object. This object is set by the `super.init(n, s)`. So, in the above example, replace:

```
clientType = s.getProperty("ips.clientType");
```

with:

```
clientType = getSession().getProperty("ips.clientType");
```

Using the Client API

After retrieving the clientType from the Session, use it to retrieve the client instance.

```
com.iplanet.portalserver.client

Class Client

java.lang.Object
  |
  +--java.util.Observable
        |
        +--com.iplanet.portalserver.client.Client
```

getInstance

```
public static Client getInstance(String clientType) throws
ClientException { }
```

The following example includes the retrieval of the clientType from the Session.

```
private String clientType = null;

public void init(String n, Session s) throws ProviderException {
    super.init(n, s);
    clientType = "";
    Client clientObj = null;
    try {
        clientType = s.getProperty("ips.clientType");
        clientObj = Client.getInstance(clientType);
    } catch (SessionException se) {
        throw new ProviderException("Unable to retrieve Session
            client type");
    } catch (ClientException ce) { }
}
```

getProperty

After establishing the client instance, use the Client `getProperty()` method to retrieve data or properties from the Client Data objects.

```
public String getProperty(String name) throws ClientException { }
```

The following example includes the:

- Retrieval of the client type from the Session
- Retrieval of the Client instance
- Retrieval of the genericHTML property value from the Client Data and setting it to a boolean

```
private String  clientType    = null;
private boolean isGenericHTML = null;
public void init(String n, Session s) throws ProviderException {
    super.init(n, s);
    clientType = "";
    Client clientObj = null;
    isGenericHTML = false;
    try {
        clientType = s.getProperty("ips.clientType");
        clientObj = Client.getInstance(clientType);
        isGenericHTML = Boolean.valueOf(clientObj.
            getProperty("genericHTML")).booleanValue();
    } catch (SessionException se) {
        throw new ProviderException("Unable to retrieve Session
            client type");
    } catch (ClientException ce) { }
}
```

getDefaultInstance

Use the Client API `getDefaultInstance()` method to retrieve the client instance corresponding to the default clientType.

```
public static Client getDefaultInstance() { }

Client clientObj = Client.getDefaultInstance();
```

getClientType

Use the Client API `getClientType()` method to retrieve the clientType string value for the current client instance.

```
public String getClientType() { }
```

The following example retrieves the default client instance and checks to see if the value of the variable clientType corresponds to the current client instance.

```
// the variable clientType has been assigned previously

if (Client.getDefaultInstance().getClientType().equals(clientType))
{
  // do something
}
```

getAllInstances

Use the Client API `getAllInstances()` method to retrieve an iterator of Client Data objects for all known client types.

```
    Iterator knownClients = Client.getAllInstances();

    Client client = null;

    while (knownClients.hasNext()) {

        try {

            client = (Client)knownClients.next();

            String clientName = client.getProperty("clientType");

            String clientUA   = client.getProperty("userAgent");

                // do something with the value

            System.out.println("Client=" + clientName);

            System.out.println("UserAgent=" + clientUA);

        } catch (ClientException ce) { }
```

```
}
```

getPropertyNames

Use the Client API `getPropertyNames()` method to retrieve a set of property names for the Client instance

```
public Set getPropertyNames() { }
```

The following example retrieves all the client instances and all the property names for each client.

```
Iterator knownClients = Client.getAllInstances();
Client client = null;
HashSet  clientProps  = null;
while (knownClients.hasNext()) {
    try {
        client = (Client)knownClients.next();
        clientProps = new HashSet(client.getPropertyNames());
        String clientName = client.getProperty("clientType");
        for (Iterator it=clientProps.iterator(); it.hasNext();) {
            System.out.println("clientProps->" + clientName + "->"
                + it.next());
        }
    } catch (Exception e) { }
}
```

profileChanged

```
public void profileChanged(ProfileEvent event) { }
```

Chapter 8

Attributes and Schemas

This chapter discusses the attributes used by the Mobile Access Pack:

- Content providers
- Authentication, Desktop, and Client Detection modules

These attributes are stored in the iPlanet Portal Server profile service. The attributes discussed in this chapter can be modified from the command line and also from the administration console graphical user interface. Please see Appendix B, "Sample Tasks" for more information on modifying these attributes.

Attributes for HTML Clients

The following HTML channel attributes are used by the server components to configure a channel for HTML clients only.

Table 8-1 Provider Attributes

Attribute	Type	Description	Acceptable Value
`<channelName>-description`	string	This attribute stores the user-visible description of the channel. This description is displayed in the Content page of the desktop.	string
`<channelName>-backgroundColor`	string	This attribute stores a valid HTML background color value for the channel	string
`<channelName>-helpLink`	string	This attribute stores the path to the channel help page for an HTML client. The path must be relative to the web server's docroot.	Relative path to the help file

Table 8-1 Provider Attributes

Attribute	Type	Description	Acceptable Value
`<channelName>-hasHelp`	boolean	This attribute specifies whether or not the channel has help	true or false
`<channelName>-column`	string	This attribute specifies the column where the channel is displayed	1-3
`<channelName>-row`	string	This attribute specifies the vertical ordering of the channel in a column	1-n
`<channelName>-width`	singlechoice	This attribute specifies the width of the channel on the user's desktop	thin, thick, full_top, or full_bottom
`<channelName>-editType`	singlechoice	This attribute specifies the type of edit page that will be returned from the channel	edit_subset or edit_complete
`<channelName>-isEditable`	boolean	This attribute specifies whether or not the channel can be edited/configured from the user's desktop	true or false
`<channelName>-hasFrame`	boolean	This attribute specifies whether or not the channel contains a frame	true or false
`<channelName>-isMovable`	boolean	This attribute specifies whether or not the channel can be moved	true or false
`<channelName>-isMinimizable`	boolean	This attribute specifies whether or not the channel can be minimized	true or false
`<channelName>-isMinimized`	boolean	This attribute specifies whether or not the channel will be displayed as just a title bar on the desktop	true or false
`<channelName>-isRemovable`	boolean	This attribute specifies whether or not the channel can be removed	true or false
`<channelName>-isDetachable`	boolean	This attribute specifies whether or not the channel can be detached	true or false
`<channelName>-isDetached`	boolean	This specifies whether or not the channel will be displayed detached from the desktop in a pop-up window	true or false
`<channelName>-hasBorder`	boolean	This attribute specifies whether or not the channel contains a border	true or false

<*channelName*>-description

The following is an example of the `<channelName>-description` attribute. This attribute stores the user-visible description of the channel. This description is displayed in the Content page.

```
<iwt:Att name="<channelName>-description"
    type="string"
    idx="a150"
    userConfigurable="false">
    <Val>Component description</Val>
    <Rperm>ADMIN</Rperm>
    <Rperm>OWNER</Rperm>
</iwt:Att>
```

<*channelName*>-backgroundColor

The following is an example of `<channelName>-backgroundColor` attribute. This attribute stores a valid HTML background color value for the channel window. HTML color values can be specified as a literal string which is the color's actual name (for example, aquamarine) or as a hexadecimal triplet (for example, #7FFFD4).

```
<iwt:Att name="<channelName>-backgroundColor"
    type="string"
    desc="Background Color"
    idx="a106"
    userConfigurable="true">
    <Val>#DDDDDD</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

<channelName>-hasHelp

The following is an example of `<channelName>-hasHelp` attribute. This attribute specifies whether or not the channel has end-user help available. A value of true indicates that the channel has help content and a value of false indicates that the channel does not include help. This is used to determine whether or not to display the help icon (?) on the channel.

```
<iwt:Priv name="<channelName>-hasHelp"
    type="boolean"
    desc="Help?"
    idx="X-x106"
    val="false">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
    <Rperm>OWNER</Rperm>
</iwt:Priv>
```

<channelName>-helpLink

The following is an example of the `<channelName>-helpLink` attribute. This attribute stores the path to the component's help page for a html client. The path must be relative to `<servername>:<portnumber>/help/<locale>`.

```
<iwt:Att name="<channelName>-helpLink"
type="string"
desc="Help Page"
idx="X-x104"
userConfigurable="true">
<Val>http://<servername>:<port>/help/<locale>/ComponentTOC.html</Va
l>
<Wperm>ADMIN</Wperm>
<Rperm>ADMIN</Rperm>
<Rperm>OWNER</Rperm>
</iwt:Att>
```

<*channelName*>-column

The following is an example of the `<channelName>-column` attribute. This attribute specifies the column in the desktop page where the channel will be displayed. Typically, the column width can be thin-thick (value not required), thick-thin (value not required), thin-thick-thin (value of 1 indicates leftmost column and value of 2 indicates rightmost column, and the thick width does not require a value), thin-thin-thin (value of 1 indicates leftmost column, value of 2 indicates the rightmost column, and value of 3 indicates the center column).

```
<iwt:Att name="<channelName>-column"

    desc="Column"

    type="string"

    idx="X-x109"

    userConfigurable="TRUE">

    <Val>1</Val>

    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>

    <Wperm>ADMIN</Wperm><Wperm>OWNER</Wperm>

</iwt:Att>
```

<*channelName*>-row

The following is an example of the `<channelName>-row` attribute. This attribute specifies the the vertical ordering or positioning of the channel in a column. If conflict arises (as a result of more than two channels requiring the same position), the order in the channel list is used.

```
<iwt:Att name="<channelName>-row"

    type="string"

    desc="Row"

    idx="X-x110"

    userConfigurable="true">

    <Val>1</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

<*channelName*>-width

The following is an example of the `<channelName>-width` attribute. This attribute specifies the width of the channel. A channel can be thin, thick, occupy the width of the desktop at the top (full_top) or botton (full_bottom).

```
<iwt:Att name="<channelName>-width"
    type="singlechoice"
    desc="Width"
    idx="X-x100"
    userConfigurable="true">
    <Val>thick</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
    <CVal>thin</CVal>
    <CVal>thick</CVal>
    <CVal>full_top</CVal>
    <CVal>full_bottom</CVal>
</iwt:Att>
```

<*channelName*>-editType

The following is an example of the `<channelName>-editType` attribute. This attribute specifies the type of edit page that will be returned for the channel. The acceptable values are:

- **edit_complete** - Indicates that it will return a complete HTML page.
- **edit_subset** - Indicates that it will return a subset of an HTML page that will be further wrapped by a template to add things like bordering, sumbit buttons, etc.

```
<iwt:Att name="<channelName>-editType"
    type="singlechoice"
    desc="Edit Form Type"
    idx=""
    userConfigurable="true">
    <Val>edit_subset</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
    <CVal>edit_subset</CVal>
    <CVal>edit_complete</CVal>
</iwt:Att>
```

<*channelName*>-isEditable

The following is an example of the `<channelName>-isEditable` attribute. This attribute specifies whether or not the channel is editable from the user's desktop. This is used to determine whether or not to display the edit icon on the channel. This is also used to control access to configuring the channel; that is, a value of false will not permit the user to configure or edit the channel setup from the user's desktop.

```
<iwt:Priv name="<channelName>-isEditable"
    type="boolean"
    desc="Editable?"
    idx="X-x107"
    val="false">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

<*channelName*>-hasFrame

The following is an example of the `<channelName>-hasFrame` attribute. This attribute specifies whether or not the channel contains a frame (see iPlanet Portal Server 3.0 Service Pack 3a Release Notes for more information on setting up framed and frameless channels).

```
<iwt:Att name="<channelName>-hasFrame"
    type="boolean"
    desc="Framed?"
    idx="X-x108"
    userConfigurable="true">
    <Val>true</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

<*channelName*>-isMovable

The following is an example of the `<channelName>-isMovable` attribute. This attribute specifies whether or not the channel can be moved in the desktop (see iPlanet Portal Server 3.0 Service Pack 3a Release Notes for more information on locking a channel's position on the desktop).

```
<iwt:Priv name="<channelName>-isMovable"
    type="boolean"
    desc="Movable?"
    idx="a107"
    val="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

<*channelName*>-isMinimizable

The following is an example of the `<channelName>-isMinimizable` attribute. This attributes specifies whether or not a channel can be minimized in the user's desktop. This is used to determine whether or not to display the Minimize icon on the channel.

```
<iwt:Priv name="<channelName>-isMinimizable"
    type="boolean"
    desc="Minimizable?"
    idx="a102"
    val="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

<*channelName*>-isMinimized

The following is an example of the `<channelName>-isMinimized` attribute. This attribute specifies whether or not the channel will be displayed as just a title bar on the desktop that the user can maximize.

```
<iwt:Att name="<channelName>-isMinimized"
    desc="Minimized?"
    type="boolean"
    idx="X-x102"
    userConfigurable="TRUE">
    <Val>false</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm><Wperm>OWNER</Wperm>
</iwt:Att>
```

<*channelName*>-isRemovable

The following is an example of the `<channelName>-isRemovable` attribute. This attribute specifies whether or not the channel can be removed from the desktop. This is used to determine whether or not to display the Remove icon on the channel.

```
<iwt:Priv name="<channelName>-isRemovable"
    type="boolean"
    desc="Removable?"
    idx="a105"
    val="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

<*channelName*>-isDetachable

The following is an example of the `<channelName>-isDetachable` attribute. This attribute specifies whether or not the channel can be detached from the desktop into a smaller pop-up window. A value of true indicates that the channel is detachable and false indicates that the channel is non-detachable.

```
<iwt:Priv name="<channelName>-isDetachable"
    type="boolean"
    desc="Detachable?"
    idx="a103"
    val="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

<*channelName*>-isDetached

The following is an example of the `<channelName>-isDetached` attribute. This attribute specifies whether the channel will be displayed detached from the desktop as a pop-up window.

```
<iwt:Att name="<channelName>-isDetached"
    type="boolean"
    desc="Detached?"
    idx="X-x101"
    userConfigurable="true">
    <Val>false</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

<*channelName*>-hasBorder

The following is an example of the `<channelName>-hasBorder` attribute. This attribute specifies whether or not the channel has a border. A true value indicates that the channel has a border and false value indicates that the channel has no border.

```
<iwt:Priv name="<channelName>-hasBorder"
    type="boolean"
    desc="Border?"
    idx="a104"
    val="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Priv>
```

Common Attributes

The following channel attributes are used by the Mobile Access Pack components to configure a channel.

Table 8-2 Provider Attributes

Attribute	Type	Description	Acceptable Value
`<channelName>-title`	string	This attribute stores theuser-visible channel title.	string
`<channelName>-clientHelpLinks`	string	This attribute stores the path to the client specific help files for the channel	clientType | path to help file
`<channelName>-refreshTime`	boolean	This attribute specifies the time interval between refreshing channel content	0 - n

<channelName>-title

The following is an example of the `<channelName>-title` attribute. This attribute stores the name or title of the channel. This is the user visible channel title. This title is displayed on the channel and in the Content page.

```
<iwt:Att name="<channelName>-title"
    desc="Title"
    type="string"
    idx="a101"
    userConfigurable="TRUE">
    <Val>Channel Name</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm>
</iwt:Att>
```

<*channelName*>-clientHelpLinks

The following is an example of the `<channelName>-clientHelpLinks` attribute. This attribute stores the path to the client specific help for the component or channel.

```
<iwt:Att name="<channelName>-clientHelpLinks"
    desc ="Component Help Links"
    type="stringlist"
    idx="a201"
    userConfigurable="TRUE">
    <Val>clientType|Path_to_help_file</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm><Wperm>OWNER</Wperm>
</iwt:Att>
```

Each clientType listed here must exactly and accurately match the clientType that is found in the Client Data. Specify only one file (such as `ComponentTOC.html` or a higher level table of contents with links to more files if needed) per component in this attribute. The path to the help file must be relative to `<iPS_install_base>/SUNWips/public_html/docs/<locale>/online_help`.

<*channelName*>-refreshTime

The following is an example of the `<channelName>-refreshTime` attribute. This attribute specifies the amount of time, in seconds, until the channel content, cached by the desktop, expires. When the user refreshes the desktop page, the Desktop checks each channel to see if the refresh time (specified in this attribute by each channel) has elapsed since the channel's content was cached. If the time had not elapsed, then the Desktop builds its page using a cached copy of the channel's content. If the cache expires, then the Desktop fetches fresh content from the channel.

```
<iwt:Att name="<channelName>-refreshTime"
    desc="Refresh Time"
    type="string"
    idx="X-x105"
    userConfigurable="TRUE">
    <Val>0</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
```

```
        <Wperm>ADMIN</Wperm>
    </iwt:Att>
```

Calendar Provider Attributes

The following attributes reside in the profile component `iwtCalendarProvider`.

Table 8-3 iwtCalendarProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtCalendarProvider-calendarUserName`	string	This attribute stores the calendar user's name.	User name
`iwtCalendarProvider-calendarUserPassword`	string	This attribute stores the calendar user's password.	User Password
`iwtCalendarProvider-calendarServerName`	string	This attribute stores the user's calendar server name.	Calendar server name
`iwtCalendarProvider-calendarServerPort`	string	This attribute stores the user's calendar server port number.	Calendar server port
`iwtCalendarProvider-weekView`	boolean	This attribute specifies whether or not the user's calendar view preference is set to weekly view.	true or false
`iwtCalendarProvider-dayView`	boolean	This attribute specifies whether or not the user's calendar view preference is set to daily view.	true or false
`iwtCalendarProvider-clientURL`	stringlist	This attribute stores the client type and the client specific calendar server URL for that client type.	clientType | CALENDAR CLIENT URL
`iwtCalendarProvider-jspLocationLength`	stringlist	This attribute specifies the number of characters about the location of the event to display.	clientType | Number of characters
`iwtCalendarProvider-jspDescLength`	stringlist	This attribute specifies the number of characters about the description of the event to display.	clientType | Number of characters
`iwtCalendarProvider-jspSummaryLength`	stringlist	This attribute specifies the number of characters about the summary of the event to display.	clientType | Number of characters
`iwtCalendarProvider-jspEventNumLines`	stringlist	This attribute specifies the number of events to display.	clientType | Number of events

iwtCalendarProvider-calendarUserName

The following is an example of the `iwtCalendarProvider-calendarUserName` attribute. This attribute's value can be specified from the edit Calendar page on the user's desktop. This attribute stores the user's calendar server id and is used to connect to the calendar server.

```
<iwt:Att name="iwtCalendarProvider-calendarUserName"
   type="string"
   desc="Calendar User Name"
   idx="a202"
   userConfigurable="true">
   <Val></Val>
   <Wperm>ADMIN</Wperm>
   <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-calendarUserPassword

The following is an example of the `iwtCalendarProvider-calendarUserPassword` attribute. This attribute's value can be specified from the edit Calendar page on the user's desktop. This attribute stores the user's calendar server password and is used to connect to the calendar server.

```
<iwt:Att name="iwtCalendarProvider-calendarUserPassword"
   type="string"
   desc="Calendar User Password"
   idx="a203"
   userConfigurable="true">
   <Val></Val>
   <Wperm>ADMIN</Wperm>
   <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-calendarServerName

The following is an example of the `iwtCalendarProvider-calendarServerName` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the calendar
- From the edit Calendar page on the user's desktop

This attribute stores the user's calendar server name and is used to connect to the user's calendar server.

```
<iwt:Att name="iwtCalendarProvider-calendarServerName"

    type="string"

    desc="Calendar Server Name"

    idx="a200"

    userConfigurable="true">

    <Val></Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtCalendarProvider-calendarServerPort

The following is an example of the `iwtCalendarProvider-calendarServerPort` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the calendar client
- From the edit Calendar page on the user's desktop

This attribute stores the user's calendar server port number and is used to connect to the user's calendar server.

```
<iwt:Att name="iwtCalendarProvider-calendarServerPort"

    type="string"

    desc="Calendar Server Port"

    idx="a201"

    userConfigurable="true">

    <Val></Val>
    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

The calendar channel uses the servername, port number, username and password set by the user in the edit page and stored in the `iwtCalendarProvider-calendarServerName`, `iwtCalendarProvider-calendarServerPort`, `iwtCalendarProvider-calendarUserName`, and `iwtCalendarProvider-calendarUserPassword` attributes respectively to connect to the calendar server and get the events.

iwtCalendarProvider-weekView

The following is an example of the `iwtCalendarProvider-weekView` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the calendar
- From the edit Calendar page on the user's desktop

If set to true, the calendar displays a weekly view of events. By default, this is set to true.

```
<iwt:Att name="iwtCalendarProvider-weekView"

    type="boolean"

    desc="Week View"

    idx="a205"

    userConfigurable="true">

    <Val>true</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtCalendarProvider-dayView

The following is an example of the `iwtCalendarProvider-dayView` attribute. This attribute gets its value:

- From the Administration Console at the time of configuring the channel for the calendar client

- From the edit Calendar page on the user's desktop

If set to true, the calendar displays a daily view of events. By default, this is set to true.

```
<iwt:Att name="iwtCalendarProvider-dayView"
    type="boolean"
    desc="Day View"
    idx="a204"
    userConfigurable="true">
    <Val>true</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-clientURL

The following is an example of the `iwtCalendarProvider-clientURL` attribute. This attribute's value can be specified from the administration console at the time of configuring the channel for the calendar client or by using the `ipsadmin` command (see "Configuring the Attributes," on page 257).

The server URL to build the Single Sign On link is configured in this profile attribute. This attribute uses a stringlist to store the clientType and the client specific server URL for that clientType. If the clientType is not found in the session, then the calendar channel defaults it to genericHTML. The channel iterates through the list of values from this attribute and gets the clientURL corresponding to the clientType accessing the calendar server.

```
<iwt:Att name="iwtCalendarProvider-clientURL"
    desc="Client URL"
    type="stringlist"
    idx="X-x200"
    userConfigurable="TRUE">
    <Val>genericHTML|INST_CALENDAR_CLIENT_URL</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm>
</iwt:Att>
```

iwtCalendarProvider-jspLocationLength

The following is an example of `iwtCalendarProvider-jspLocationLength` attribute. This attribute specifies the number of characters to display about the event location on the corresponding client. This is only applicable to clients that utilize the calendar JSP template files (discussed in "iPlanet Calendar Server JSPs," on page 60).

```
<iwt:Att name="iwtCalendarProvider-jspLocationLength"
    type="stringlist"
    desc="Length of Event Location to Display Via JSP"
    idx="X-x113"
    userConfigurable="true">
    <Val>genericHTML|40</Val>
    <Val>cHTML|25</Val>
    <Val>WML|25</Val>
    <Val>nokia_7110|25</Val>
    <Val>siemans_s35|25</Val>
    <Val>pana_p503i|25</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-jspDescLength

The following is an example of `iwtCalendarProvider-jspDescLength` attribute. This attribute specifies the number of characters to display about the event description on the corresponding client. This is only applicable to clients that utilize the calendar JSP template files (discussed in "iPlanet Calendar Server JSPs," on page 60).

```
<iwt:Att name="iwtCalendarProvider-jspDescLength"
    type="stringlist"
    desc="Length of Event Description to Display Via JSP"
    idx="X-x114"
    userConfigurable="true">
    <Val>genericHTML|1024</Val>
    <Val>cHTML|300</Val>
    <Val>WML|300</Val>
    <Val>nokia_7110|300</Val>
    <Val>siemans_s35|300</Val>
    <Val>pana_p503i|300</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-jspSummaryLength

The following is an example of the `iwtCalendarProvider-jspSummaryLength` attribute. This attribute specifies the number of characters to display about the event summary on the corresponding client. This is only applicable to clients that utilize the calendar JSP template files (discussed in "iPlanet Calendar Server JSPs," on page 60).

```
<iwt:Att name="iwtCalendarProvider-jspSummaryLength"
    type="stringlist"
    desc="Length of Event Summary to Display Via JSP"
    idx="X-x112"
    userConfigurable="true">
    <Val>genericHTML|40</Val>
    <Val>cHTML|25</Val>
    <Val>WML|25</Val>
    <Val>nokia_7110|25</Val>
    <Val>siemans_s35|25</Val>
    <Val>pana_p503i|25</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtCalendarProvider-jspEventNumLines

The following is an example of the `iwtCalendarProvider-jspEventNumLines` attribute. This attribute stores the number of calendar events to display on the corresponding client. This is only applicable to clients that utilize the calendar JSP template files (discussed in "iPlanet Calendar Server JSPs," on page 60).

```
<iwt:Att name="iwtCalendarProvider-jspEventNumLines"
    type="stringlist"
    desc="Number of Events to Display Via JSP"
    idx="X-x111"
    userConfigurable="true">
    <Val>genericHTML|10</Val>
    <Val>cHTML|7</Val>
    <Val>WML|7</Val>
    <Val>nokia_7110|7</Val>
    <Val>siemans_s35|7</Val>
    <Val>pana_p503i|7</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

Mail Provider Attributes

The attributes discussed in Table 8-4 reside in the profile component `iwtMailProvider`.

Table 8-4 iwtMailProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtMailProvider-msgCopyInSent`	boolean	This attribute specifies whether or not the user prefers to store a copy of the message in the Sent folder	true or false
`iwtMailProvider-clientURL`	stringlist	This attribute stores the URL to the mail server for each supported client type	*clientType | URL_to_Mail_Server*

Table 8-4 iwtMailProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtMailProvider-clientPort`	string	This attribute stores the mail client port number	clientPortNumber
`iwtMailProvider-sortOrder`	stringlist	This attribute specifies the order to sort the messages by	top or bottom
`iwtMailProvider-displayHeaders`	boolean	This attribute specifies whether or not to display the message headers on the user's mail channel	true or false
`iwtMailProvider-numberHeaders`	string	This attribute specifies the number of message headers to display on the user's mail channel	0 to n
`iwtMailProvider-predefinedReplies`	stringlist	This attribute stores the user's predefined replies	Predefined replies
`iwtMailProvider-signature`	string	This attribute stores the user's signature	User's signature
`iwtMailProvider-changeUserInfo`	boolean	This attribute specifies whether or not to update the user profile	true or false
`iwtMailProvider-IMAPServerName`	string	This attribute stores the user's IMAP server name	Server Name
`iwtMailProvider-IMAPUserId`	string	This attribute stores the user's IMAP user id	User Id
`iwtMailProvider-IMAPPassword`	protected	This attribute stores the user's IMAP password	User's Password
`iwtMailProvider-jspInboxNumLines`	stringlist	This attribute specifies the number of lines from the Inbox to display	clientType|number of lines
`iwtMailProvider-jspFromLength`	stringlist	This attribute specifies the number of characters from the From header to display	clientType|number of characters
`iwtMailProvider-jspSubjectLength`	stringlist	This attribute specifies the number of characters from the Subject header to display	clientType|number of characters
`iwtMailProvider-jspMsgNumLines`	stringlist	This attribute specifies the number of lines from the message to display	clientType|number of lines

iwtMailProvider-msgCopyInSent

The following is an example of the `iwtMailProvider-msgCopyInSent` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the mail client
- From the edit Mail page on the user's desktop

This attribute indicates whether or not to save a copy of the message in the user's Sent folder. By default, this is set to true.

```
<iwt:Att name="iwtMailProvider-msgCopyInSent"

    type="boolean"

    desc="Place a copy of message in Sent"

    idx="a206"

    userConfigurable="true">

    <Val>true</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtMailProvider-clientURL

The following is an example of the `iwtMailProvider-clientURL` attribute. This attribute's value can be specified from the administration console at the time of configuring the channel for the mail client and/or from the Edit Mail channel page.

The server URL to build the Single Sign On link is configured in this profile attribute. This attribute uses a stringlist to store the clientType and the client specific server URL for that clientType. If the client type is not found in the session, then the mail channel defaults it to generic HTML. The channel iterates through the list of values from this attribute and gets the client URL corresponding to the clientType accessing the mail server.

```
<iwt:Att name="iwtMailProvider-clientURL"

    type="stringlist"

    desc="Client URL"

    idx="a203"

    userConfigurable="true">
    <Val>genericHTML|INST_MAIL_CLIENT_URL</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-clientPort

The following is an example of the `iwtMailProvider-clientPort` attribute. This attribute stores the port number of the mail client.

```
<iwt:Att name="iwtMailProvider-clientPort"
    type="string"
    desc="Client Port"
    idx="a207"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-sortOrder

The following is an example of the `iwtMailProvider-sortOrder` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the mail client
- From the edit Mail page on the user's desktop

This attribute indicates the order in which to sort the user's messages in the inbox for display. The user can either view the most recent at the top or at the bottom. By default, the most recent message are displayed at the top of the inbox.

```
<iwt:Att name="iwtMailProvider-sortOrder"
    type="string"
    desc="Sort Order"
    idx="a202"
    userConfigurable="true">
    <Val>top</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-displayHeaders

The following is an example of the `iwtMailProvider-displayHeaders` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the mail client
- From the edit Mail page on the user's desktop

This attribute indicates whether or not to display the message headers in the user's mail channel. By default, this is set to true.

```
<iwt:Att name="iwtMailProvider-displayHeaders"
    type="string"
    desc="Display Headers"
    idx="a200"
    userConfigurable="true">
    <Val>true</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-numberHeaders

The following is an example of the `iwtMailProvider-numberHeaders` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the mail client
- From the edit Mail page on the user's desktop

This attribute indicates the number of message headers to display in the user's mail channel. By default, up to five message headers are displayed in the user's mail channel and the channel limits the maximum number of message headers to be displayed to 30.

```
<iwt:Att name="iwtMailProvider-numberHeaders"
    type="string"
    desc="Number of Headers"
    idx="a201"
    userConfigurable="true">
    <Val>5</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-predefinedReplies

The following is an example of the `iwtMailProvider-predefinedReplies` attribute. This attribute's value can be specified from the edit Mail page on the user's desktop and/or from the Administration Console. This attribute allows the user to create and store messages to send out later, typically, from the mobile device. The predefined messages cannot exceed thirty characters. Although these messages can be set up from the Administration Console as well, these messages must be created by the user from their edit Mail page.

```
<iwt:Att name="iwtMailProvider-predefinedReplies"
    type="stringlist"
    desc="Predefined Replies"
    idx="a205"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-signature

The following is an example of the `iwtMailProvider-signature` attribute. This attribute's value can be specified from the edit Mail page on the user's Desktop and/or from the Administration Console. This attribute allows the user to create and store a signature from the user's Desktop. Although this can be set from the administration console as well, this must be set by the user from the edit Mail page.

```
<iwt:Att name="iwtMailProvider-signature"
    type="string"
    desc="Signature"
    idx="a204"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-changeUserInfo

The following is an example of the `iwtMailProvider-changeUserInfo` attribute. This attribute specifies whether or not to update the user's IMAP server name, user id, and password information in the user info channel. If this attribute is set to true, the IMAP server name (see `iwtMailProvider-IMAPServerName`), user id (see `iwtMailProvider-IMAPUserId`), and password (see `iwtMailProvider-IMAPPassword`) specified via this component will overwrite the IMAP server name, user id, and password specified in the user info channel.

```
<iwt:Att name="iwtMailProvider-changeUserInfo"
    type="boolean"
    desc="Change Mail Server attributes on UserInfo"
    idx="a211"
    userConfigurable="true">
    <Val>false</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-IMAPServerName

The following is an example of the `iwtMailProvider-IMAPServerName` attribute. This attribute stores the user's IMAP server name.

```
<iwt:Att name="iwtMailProvider-IMAPServerName"
    type="string"
    desc="IMAP ServerName to authenticate against"
    idx="a208"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-IMAPUserId

The following is an example of the `iwtMailProvider-IMAPUserId` attribute. This attribute stores the user's IMAP server user id.

```
<iwt:Att name="iwtMailProvider-IMAPUserId"
    type="string"
    desc="Mail Server User Id"
    idx="a209"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-IMAPPassword

The following is an example of the `iwtMailProvider-IMAPPassword` attribute. This attribute stores the user's IMAP server password.

```
<iwt:Att name="iwtMailProvider-IMAPPassword"
    type="protected"
    desc="Mail Server User Password"
    idx="a210"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
```

```
</iwt:Att>
```

iwtMailProvider-jspInboxNumLines

The following is an example of the `iwtMailProvider-jspInboxNumLines` attribute. This attribute specifies the number of lines from the Inbox to display on the corresponding client. This is only applicable to clients that utilize the mail JSP templates.

```
<iwt:Att name="iwtMailProvider-jspInboxNumLines"
    type="stringlist"
    desc="Number of Inbox Lines to Display Via JSP"
    idx="X-x111"
    userConfigurable="true">
    <Val>genericHTML|10</Val>
    <Val>cHTML|11</Val>
    <Val>WML|9</Val>
    <Val>nokia_7110|9</Val>
    <Val>siemans_s35|9</Val>
    <Val>pana_p503i|10</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-jspFromLength

The following is an example of the `iwtMailProvider-jspFromLength` attribute. This attribute specifies the number of characters from the From header to display on the corresponding client. This is only applicable to clients that utilize the mail JSP templates.

```
<iwt:Att name="iwtMailProvider-jspFromLength"
    type="stringlist"
    desc="Length of From Header to Display Via JSP"
    idx="X-x113"
    userConfigurable="true">
    <Val>genericHTML|40</Val>
    <Val>cHTML|18</Val>
    <Val>WML|23</Val>
    <Val>nokia_7110|23</Val>
    <Val>siemans_s35|23</Val>
    <Val>pana_p503i|18</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-jspSubjectLength

The following is an example of the `iwtMailProvider-jspSubjectLength` attribute. This attribute specifies the number of characters from the Subject header to display on the corresponding client. This is only applicable to clients that utilize the mail JSP template files.

```
<iwt:Att name="iwtMailProvider-jspSubjectLength"
    type="stringlist"
    desc="Length of Subject Header to Display Via JSP"
    idx="X-x114"
    userConfigurable="true">
    <Val>genericHTML|40</Val>
    <Val>cHTML|18</Val>
    <Val>WML|23</Val>
    <Val>nokia_7110|23</Val>
    <Val>siemans_s35|23</Val>
    <Val>pana_p503i|18</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtMailProvider-jspMsgNumLines

The following is an example of the `iwtMailProvider-jspMsgNumLines` attribute. This attribute specifies the number of lines from the message to display on the corresponding client. This is only applicable to clients that utilize the mail JSP template files.

```
<iwt:Att name="iwtMailProvider-jspMsgNumLines"
    type="stringlist"
    desc="Number of Message Lines to Display Via JSP"
    idx="X-x112"
    userConfigurable="true">
    <Val>genericHTML|10</Val>
    <Val>cHTML|11</Val>
    <Val>WML|9</Val>
    <Val>nokia_7110|9</Val>
    <Val>siemans_s35|9</Val>
    <Val>pana_p503i|11</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

Address Book Provider Attributes

The attributes discussed in Table 8-5 reside in the profile component `iwtAddressBookProvider`.

Table 8-5 iwtAddressBookProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtAddressBookProvider -clientURL`	stringlist	This attribute stores the URL to the iPlanet Messaging Server hosting the address book for each supported client type	clientType|UR L_TO_AB_SER VER
`iwtAddressBookProvider -displayEntries`	string	This attribute indicates whether or not to display the address book entries	true or false

Table 8-5 iwtAddressBookProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtAddressBookProvider -sortOrder`	string	This attribute specifies the sor order for the address book entries	up or down
`iwtAddressBookProvider -sortBy`	string	This attribute specifies the field to sort the address book entries by	firstname or lastname
`iwtAddressBookProvider -numEntries`	string	This attribute specifies the number of address book entries to display	0 - n
`iwtAddressBookProvider -maxEntries`	string	This attribute specifies the maximum number of address book entries to display	0 - n
`iwtAddressBookProvider -LDAPServerName`	string	This attribute stores the address book database server name	LDAP server name
`iwtAddressBookProvider -LDAPport`	string	This attribute stores the address book database server port number	LDAP server port
`iwtAddressBookProvider -LDAPAdminId`	string	This attribute stores the distinguished name of the address book database administrator	LDAP admin DN
`iwtAddressBookProvider -LDAPAdminPassword`	string	This attribute stores the password of the address book database administrator	LDAP admin password
`iwtAddressBookProvider -LDAPSearchBase`	string	This attribute stores the base distinguished name in the LDAP database for address book	LDAP search base
`iwtAddressBookProvider -IMAPServerName`	string	This attribute stores the user's IMAP server name	IMAP server name
`iwtAddressBookProvider -IMAPUserId`	string	This attribute stores the user's IMAP user id	IMAP user id
`iwtAddressBookProvider -IMAPPassword`	protected	This attribute stores the user's IMAP password.	IMAP user password
`iwtAddressBookProvider -ldapConnPoolStart`	stringlist	This attribute specifies the number to LDAP connection pools to start.	0 - n
`iwtAddressBookProvider -ldapConnPoolMax`	stringlist	This attribute specifies the maximum number to LDAP connection pools to establish.	0 - n
`iwtAddressBookProvider -ldapOpTimeout`	stringlist	This attribute specifies the time, in milliseconds, after which the LDAP operation will terminate.	time in milliseconds

Table 8-5 iwtAddressBookProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtAddressBookProvider -jspAbNumLines`	stringlist	This attribute specifies the number of address book entries to display	clientType | nu mber of lines

iwtAddressBookProvider-clientURL

The following is an example of the `iwtAddressBookProvider-clientURL` attribute. This attribute's value can be specified from the administration console at the time of configuring the client channel.

The server URL used to build the Single Sign On link is configured in this profile attribute. This attribute uses a stringlist to store the clientType and the client specific server URL for that clientType. If the clientType is not found in the session, then the address book channel defaults it to generic HTML. The channel iterates through the list of values from this profile component attribute and gets the client URL based on the clientType accessing the address book server.

The `isPresentable()` method tries to get the client specific template for the channel content. If the template exists, then it returns true, else returns false.

```
<iwt:Att name="iwtAddressBookProvider-clientURL"

    type="stringlist"

    desc="Client URL"

    idx="a203"

    userConfigurable="true">

    <Val>genericHTML|INST_AB_CLIENT_URL</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtAddressBookProvider-displayEntries

The following is an example of the `iwtAddressBookProvider-displayEntries` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the address book client
- From the edit Address Book page on the user's desktop

This attribute indicates whether or not to display the user's address book entries in the address book channel. By default, it is set to true.

```
<iwt:Att name="iwtAddressBookProvider-displayEntries"
    type="string"
    desc="Display Entries"
    idx="a300"
    userConfigurable="true">
    <Val>true</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-sortOrder

The following is an example of the `iwtAddressBookProvider-sortOrder` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the address book client
- From the edit Address Book page on the user's desktop

This attribute indicates the order in which to sort the user's address book entries for display. The user can either select an ascending order or descending order of display of entries. By default, the address book entries are sort in an ascending order. The possible values are `up` and `down`.

```
<iwt:Att name="iwtAddressBookProvider-sortOrder"
    type="string"
    desc="Sort Order"
    idx="a304"
    userConfigurable="true">
    <Val>up</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-sortBy

The following is an example of the `iwtAddressBookProvider-sortBy` attribute. This attribute gets its value:

- From the administration console at the time of configuring the channel for the address book client
- From the edit Address Book page on the user's desktop

This attribute indicates the field in which to sort the user's address book entries for display. The user can either select to display entries by first name or last name. By default, the address book entries are sort by first name. The possible values are `firstname` and `lastname`.

```
<iwt:Att name="iwtAddressBookProvider-sortBy"
    type="string"
    desc="Sort on given field"
    idx="a303"
    userConfigurable="true">
    <Val>Lastname</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-numEntries

The following is an example of the `iwtAddressBookProvider-numEntries` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the channel for the address book client
- From the edit Address Book page on the user's desktop

This attribute specifies the number of address book entries to display on the user's address book channel. By default, up to five entries will be displayed on the user's address book channel.

Note that the number of entries that can be displayed is limited by the maximum number of entries allowed to be displayed on the channel (see `iwtAddressBookProvider-maxEntries` for more information).

```
<iwt:Att name="iwtAddressBookProvider-numEntries"
```

```
    type="string"
    desc="Number of Entries"
    idx="a301"
    userConfigurable="true">
    <Val>5</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-maxEntries

The following is an example of the `iwtAddressBookProvider-maxEntries` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the channel for the address book client. This attribute indicates the maximum number of entries that can displayed on the address book channel. By default, up to thirty entries can be displayed on the user's address book channel.

```
<iwt:Att name="iwtAddressBookProvider-maxEntries"
    type="string"
    desc="Maximum number of Entries"
    idx="a302"
    userConfigurable="true">
    <Val>30</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-LDAPServerName

The following is an example of the `iwtAddressBookProvider-LDAPServerName` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the channel for the address book client. This attribute stores the address book database server name. This is used at the time of retrieving entries for the user and for performing searches in the address book database.

```
<iwt:Att name="iwtAddressBookProvider-LDAPServerName"
    type="string"
    desc="Address Book LDAP server"
    idx="1200"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-LDAPport

The following is an example of the `iwtAddressBookProvider-LDAPport` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the channel for the address book client. This attribute stores the address book database server port number. This is used at the time of retrieving entries for the user and for performing searches in the address book database.

```
<iwt:Att name="iwtAddressBookProvider-LDAPport"
    type="string"
    desc="Address Book LDAP server port"
    idx="1201"
    userConfigurable="true">
    <Val>389</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-LDAPAdminId

The following is an example of the `iwtAddressBookProvider-LDAPAdminId` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the channel for the address book client. This attribute stores the address book database administrator's name. This is used at the time of retrieving entries for the user and for performing searches in the address book database.

```
<iwt:Att name="iwtAddressBookProvider-LDAPAdminId"
   type="string"
   desc="Address Book Admin user DN"
```

```
    idx="1202"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-LDAPAdminPassword

The following is an example of the `iwtAddressBookProvider-LDAPAdminPassword` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the channel for the address book client. This attribute stores the address book database administrator's password. This is used at the time of retrieving entries for the user and for performing searches in the address book database.

```
<iwt:Att name="iwtAddressBookProvider-LDAPAdminPassword"
    type="string"
    desc="Address Book Admin user password"
    idx="1203"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-LDAPSearchBase

The following is an example of the `iwtAddressBookProvider-LDAPSearchBase` attribute. This attribute gets its value from the Administration Console at the time of configuring the channel for the address book client. This attribute specifies the directory search base in which to initiate the search for the LDAP directory.

```
<iwt:Att name="iwtAddressBookProvider-LDAPSearchBase"
    type="string"
    desc="Searchbase for the LDAP directory"
    idx="1204"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-IMAPServerName

The following is an example of the `iwtAddressBookProvider-IMAPServerName` attribute. This attribute's value can be specified from the Administration Console at the time of configuring the address book client channel. This attribute stores the user's IMAP server name. This is used for authentication of the Address Book users.

```
<iwt:Att name="iwtAddressBookProvider-IMAPServerName"
    type="string"
    desc="IMAP ServerName to authenticate against"
    idx="a200"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-IMAPUserId

The following is an example of the `iwtAddressBookProvider-IMAPUserId` attribute. This attribute gets its value from the edit Address Book page on the user's desktop. This attribute stores the user's IMAP user id. This is used to verify the validity of the credentials of the user against the specified IMAP server.

```
<iwt:Att name="iwtAddressBookProvider-IMAPUserId"
    type="string"
    desc="Address Book User Id"
    idx="a201"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-IMAPPassword

The following is an example of the `iwtAddressBookProvider-IMAPPassword` attribute. This attribute stores the user's IMAP user password. This is used to verify the validity of the credentials of the user against the specified IMAP server.

```
<iwt:Att name="iwtAddressBookProvider-IMAPPassword"
    type="protected"
    desc="Address Book User Password"
    idx="a202"
    userConfigurable="true">
    <Val></Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtAddressBookProvider-ldapConnPoolStart

The following is an example of the `iwtAddressBookProvider-ldapConnPoolStart` attribute. This attribute can be modified only by using the `ipsadmin` command (see "Configuring the Attributes," on page 257). This attribute specifies the number of LDAP connections to open initially in the connection pool used to connect to the LDAP server handling the address book. By default, upto five connections are established.

```
<iwt:Att name="iwtAddressBookProvider-ldapConnPoolStart"
    type="string"
    desc="The number of LDAP connections to start off with"
    idx="g100"
    userConfigurable="false">
    <Val>5</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
```

```
</iwt:Att>
```

iwtAddressBookProvider-ldapConnPoolMax

The following is an example of the `iwtAddressBookProvider-ldapConnPoolMax` attribute. This attribute can be modified only by using the `ipsadmin` command (see "Configuring the Attributes," on page 257). This attribute specifies the maximum number of LDAP connections to allow in the connection pool used to connect to the LDAP server handling the address book. By default, a maximum of twenty connections are allowed.

```
<iwt:Att name="iwtAddressBookProvider-ldapConnPoolMax"

    type="string"

    desc="Maximum number of LDAP connections in the connection pool"

    idx="g101"

    userConfigurable="false">

    <Val>20</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtAddressBookProvider-ldapOpTimeout

The following is an example of the `iwtAddressBookProvider-ldapOpTimeout` attribute. This attribute can be modified only by using the `ipsadmin` command (see "Configuring the Attributes," on page 257). This attribute specifies the amount of time, in milliseconds, after which the LDAP operation to the ldap server must terminate if it does not receive a response back from the server. By default, the LDAP operation will terminate after 20000 ms (20 seconds).

```
<iwt:Att name="iwtAddressBookProvider-ldapOpTimeout"

    type="string"

    desc="The time in ms after which an ldap operation times out"

    idx="g102"

    userConfigurable="false">

    <Val>20000</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>
```

```
</iwt:Att>
```

iwtAddressBookProvider-jspAbNumLines

The following is an example of the `iwtAddressBookProvider-jspAbNumLines` attribute. This attribute specifies the number of address book lines (entries) to display on the corresponding client. This is only applicable to clients that utilize the address book JSP template files.

```
<iwt:Att name="iwtAddressBookProvider-jspAbNumLines"
    type="stringlist"
    desc="Number of AB Lines to Display Via JSP"
    idx="X-x111"
    userConfigurable="true">
    <Val>genericHTML|10</Val>
    <Val>cHTML|9</Val>
    <Val>WML|9</Val>
    <Val>nokia_7110|9</Val>
    <Val>siemans_s35|9</Val>
    <Val>pana_p503i|9</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

Bookmark Provider Attributes

The attributes discussed in Table 8-6 reside in the profile component `iwtWirelessBookmarkProvider`.

Table 8-6 iwtWirelessBookmarkProvider* attributes

Attribute	Type	Description	Acceptable Value
iwtWirelessBookmarkProviderWML-targets	stringlist	This attribute stores the resources or URLs for a WML client	URL
iwtWirelessBookmarkProvidercHTML-targets	stringlist	This attribute stores the resources or URLs for a cHTML client	URL

iwtWirelessBookmarkProviderWML-targets

The following is an example of the `iwtWirelessBookmarkProviderWML-targets` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the client specific bookmark channel
- From the edit WML Bookmark page on the user's desktop

This attribute stores the bookmarked URLs or resources for a WML client.

```
<iwt:Att name="iwtWirelessBookmarkProviderWML-targets"
    type="stringlist"
    desc="Boomarks"
    idx="a200"
    userConfigurable="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtWirelessBookmarkProvidercHTML-targets

The following is an example of the `iwtWirelessBookmarkProvidercHTML-targets` attribute. This attribute's value can be specified:

- From the Administration Console at the time of configuring the client specific bookmark channel
- From the edit cHTML Bookmark page on the user's desktop

This attribute stores the bookmarked URLs or resources for a cHTML client.

```
<iwt:Att name="iwtWirelessBookmarkProvidercHTML-targets"
    type="stringlist"
    desc="Boomarks"
    idx="a200"
    userConfigurable="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

XML Provider Attributes

The attributes discussed in Table 8-7 reside in the profile component `iwtXMLProvider`.

Table 8-7 iwtXMLProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtXMLProvider-xslCache`	boolean	This attribute indicates whether or not caching is enabled. By default, it is enabled.	true or false
`iwtXMLProvider-url`	string	This attributes stores the location of the XML content file or the URL that generates the XML content. This attribute can have URLs of type of HTTP or HTTPS or File URL.	Path to XML source
`iwtXMLProvider-xslFileName`	string	This attribute stores the XSL template file to be used for transformation.	Absolute path (ex. /abc/xyz/test.xsl) to the template file or the base filename only

iwtXMLProvider-xslCache

The following is an example of the `iwtXMLProvider-xslCache` attribute. This attribute specifies whether or not caching of pre-processed XSLs to use for transformation is enabled. By default, this is enabled.

```
<iwt:Att name="iwtXMLProvider-xslCache"
    desc="XSL Cache ?"
    type="boolean"
    idx="X-x301"
    userConfigurable="TRUE">
    <Val>true</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm>
</iwt:Att>
```

iwtXMLProvider-url

The following is an example of the `iwtXMLProvider-url` attribute. This attribute specifies the location of the XML content file or the URL that generates the XML content. This attribute can store:

- URLs of type of HTTP or HTTPS
- File URL

```
<iwt:Att name="iwtXMLProvider-url"
    desc="XML Content URL"
    type="string"
    idx="a200"
    userConfigurable="TRUE">
    <Val>http://mum.red.iplanet.com:8080/books.xml</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm>
</iwt:Att>
```

iwtXMLProvider-xslFileName

The following is an example of the `iwtXMLProvider-xslFileName` attribute. This attribute specifies the XSL template file to be used for transforming the XML content. It can store:

- The absolute path to the XSL template file
- The base XSL filename. If this is specified, the path will be determined using the file lookup mechanism.

```
<iwt:Att name="iwtXMLProvider-xslFileName"
    desc="XSL Filename"
    type="string"
    idx="X-x109"
    userConfigurable="TRUE">
    <Val>books1.xsl</Val>
    <Rperm>ADMIN</Rperm><Rperm>OWNER</Rperm>
    <Wperm>ADMIN</Wperm>
</iwt:Att>
```

See the Administration Guide for information on saving XSL template files.

JAXP Jar Attributes

The attributes discussed in Table 8-7 reside in the profile component `iwtJAXPJars`.

Table 8-8 iwtJAXPJars Attributes

Attribute	Type	Description	Acceptable Value
`iwtJAXPJars-jaxpJars`	stringlist	This attribute stores the location of the JAXP (Java API for XML Parsing v1.1) jar files.	Path to `crimson.jar`, `xalan.jar`, and `jaxp.jar`

The following is an example of the `iwtJAXPJars-jaxpJars` attribute. This attribute stores the path to the JAXP jar files. This includes (and requires) the relative path from the directory where the iPlanet Portal Server server is installed (which is, by default, `/opt`). These files are used by the XML Provider for the XSLT engine and these files are used by all the XMLprovider channels.

```
<iwt:Att name="iwtJAXPJars-jaxpJars"
    type="stringlist"
    desc="Location of JAXP Jar Files relative to install dir."
    idx="X-x1"
    userConfigurable="false">
    <Val>SUNWips/lib/xmlp/crimson.jar</Val>
    <Val>SUNWips/lib/xmlp/xalan.jar</Val>
    <Val>SUNWips/lib/xmlp/jaxp.jar</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

Wireless URLScraper Provider Attributes

The attributes discussed in Table 8-9 reside in the profile component `iwtWirelessURLScraperProvider`.

Table 8-9 iwtWirelessURLScraperProvider Attributes

Attribute	Type	Description	Acceptable Value
`iwtWirelessURLScraperProvider-url`	stringlist	This attribute stores the default Content URL.	URL_for_html_clients
`iwtWirelessURLScraperProvider-showAsLink`	boolean	This attribute displays the URL as link for non-desktop browsers instead of scraping.	true or false
`iwtWirelessURLScraperProvider-clientPresentableTo`	stringlist	This attribute stores the list of clients that can use the default URL.	clientType
`iwtWirelessURLScraperProvider-urlList`	stringlist	This attribute stores the list of URLs for clients that can support the default URL.	clientType | URL

iwtWirelessURLScraperProvider-url

The following is an example of the `iwtWirelessURLScraperProvider-url` attribute. This attribute stores the URL from which to scrape the contents for display on a HTML client. This default content URL is used for serving up content to HTML browsers. In order to serve this URL to other clients, add the list of clients to serve this URL to in the `iwtWirelessURLScraperProvider-clientPresentableTo` attribute.

```
<iwt:Att name="iwtWirelessURLScraperProvider-url"
    type="string"
    desc="Default Content URL"
    idx="a200"
    userConfigurable="true">
    <Val>http://www.iplanet.com</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtWirelessURLScraperProvider-showAsLink

The following is an example of the `iwtWirelessURLScraperProvider-showAsLink` attribute. This attribute specifies whether or not to display the URL as a link on a mobile device instead of scraping and displaying the contents of the URL on the mobile device.

On a HTML or a cHTML client, the behaviour of the WirelessURLScraper is similar to the iPlanet Portal Server URLScraper. However, on a WML client, the WirelessURLScraper can display the contents of only the first card for WML pages, if the URL contains more than one card.

Since a mobile device has a small amount of memory and display area and since the scraped contents cannot have multiple cards, this attribute provides an option to display a link instead of scraping the contents in the URL.

```
<iwt:Att name="iwtWirelessURLScraperProvider-showAsLink"
    type="boolean"
    desc="Display the URL as a link for non-desktop browsers"
    idx="a206"
    userConfigurable="true">
    <Val>true</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

The WirelessURLScraper uses the `link.template` file in the `filePath` attribute (defined for the clientType in `iwtClient-clientTypes`) under either:

1. `<template_dir>/[channelName_as_provided_in_Wizard]/<clientType>`

   Or

2. `<template_dir>/iwtWirelessURLScraperprovider/<clientType>`

It uses the `link.template` file it first encounters.

iwtWirelessURLScraperProvider-clientPresentableTo

The following is an example of the `iwtWirelessURLScraperProvider-clientPresentableTo` attribute. The default content URL is used for serving content to HTML browsers. If this URL can also be served to other HTML and non-HTML clients, add the clients in this attribute.This attribute stores the list of clients that can use the default content URL. If the value of this attribute is null (or if no client is listed in this attribute), all clients can use the default content URL. However, if this attribute contains some value (client), then only the listed clients can use the default content URL.

```
<iwt:Att name="iwtWirelessURLScraperProvider-clientPresentableTo"
    type="stringlist"
    desc="Clients that can use default URL"
    idx="a207"
    userConfigurable="true">
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtWirelessURLScraperProvider-urlList

The following is an example of the `iwtWirelessURLScraperProvider-urlList` attribute. The `iwtWirelessURLScraperProvider-urlList` attribute stores the default URLs for all of the supported clientTypes. When a client makes a request, the `isPresentable()` method looks-up the URL configured for the requesting clientType. If a URL is found, the scraper will return true, or false otherwise.

```
<iwt:Att name="iwtWirelessURLScraperProvider-urlList"
   type="stringlist"
   desc="List of URLs for various devices"
   idx="a205"
   userConfigurable="true">
   <Val>wml|http://wap.iplanet.com</Val>
   <Val>wml|http://wap.sun.com</Val>
   <Val>Nokia|http://wap.iplanet.com.nk</Val>
   <Val>Nokia7110|http://wap.sun.com.nk7110</Val>
   <Wperm>ADMIN</Wperm>
   <Rperm>ADMIN</Rperm>
</iwt:Att>
```

The URLScraper uses the `iwtClient-clientTypes` attribute in the profile to retrieve the URL most suitable for the client, based on the filePath for the particular client. For example, if the request comes from a Nokia 7110 phone, the scraper fetches the filePath from the `iwtClient-clientTypes` (such as `filePath=wml/Nokia/Nokia7110`) and looks in the `iwtWirelessURLScraperProvider-urlList` for the page most suitable for this device. It looks for Nokia7110, followed by Nokia, and falls back to WML in case no specific URL is available. Note that the search is case-sensitive.

Authentication Module Attributes

The attributes discussed in Table 8-10 reside in the profile component `iwtAuth`.

Table 8-10 iwtAuth Attributes

Attribute	Type	Description	Acceptable Value
`iwtAuth-clientDetectionClass`	string	This attribute stores the name of the class implementing the ClientDetectionInterface.	com.iplanet.portalserver.client.ClientDetector
`iwtAuth-clientDetectionEnabled`	boolean	This attribute determines whether or not the authentication service will perform Client Detection. By default, this attribute is enabled.	true or false
`iwtAuth-supportedAuthModules`	stringlist	This attribute stores the modules supported by each supported clientType.	clientType | Supported_Auth Modules
`iwtAuth-loginWorkerClass`	stringlist	This attribute stores the loginWorkerClass for HTML clients.	default | com.iplanet.portalserver.auth.server.HTMLLoginWorker
`iwtAuth-loginWorkerClasses`	stringlist	This attribute identifies which clientType should use which loginWorker class to generate the authentication pages.	clientType | LoginWorkerClassPackage

iwtAuth-clientDetectionClass

The following is an example of the `iwtAuth-clientDetectionClass` attribute. This attribute stores the name of the class implementing the client detection interface. By default, `com.iplanet.portalserver.client.ClientDetector` is the name of the client detection interface.

```
<iwt:Att name="iwtAuth-clientDetectionClass"

    type="string"

    desc="Client Detector Class"

    idx="X-x28"

    userConfigurable="false">

    <Val>com.iplanet.portalserver.client.ClientDetector</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtAuth-clientDetectionEnabled

The following is an example of the `iwtAuth-clientDetectionEnabled` attribute. This attribute indicates whether or not client detection is enabled for authentication. That is, this indicates whether or not to detect the clientType accessing the Mobile Access Pack server at the time of authenticating the client to the server.

The clientType is determined before any content is returned to the requesting client and the server looks up the contentType that the client accepts. Other components also use the clientType value as an index in the Client Data to retrieve information about the client accessing the server. By default, this is enabled.

```
<iwt:Att name="iwtAuth-clientDetectionEnabled"

    type="boolean"

    desc="Client Detection Enabled"

    idx="X-x27"

    userConfigurable="false">

    <Val>true</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

To determine the client type, the ClientDetectionInterface will:

- Determine the client type based on the request
- Send a HTTPServletRequest
- Return a String representing the clientType
- If there is an error retrieving the client data, throw a ClientDetectionException

iwtAuth-supportedAuthModules

The following is an example of the `iwtAuth-supportedAuthModules` attribute. This attribute stores the modules supported by a particular client as a string list with each value in the list having the clientType and the modules it supports. This is to enable the authentication service to identify which authentication modules are supported by a particular client. Based on the stringlist stored in this attribute, the authentication service displays only those module names which are supported by the requesting client in the authentication menu for the client.

```
<iwt:Att name="iwtAuth-supportedAuthModules"
    type="stringlist"
    desc="Supported Auth Modules for Clients"
    idx="X-x29"
    userConfigurable="false">
   <Val>default|Unix;Radius;Ldap</Val>
   <Val>WML|Unix;Radius;Ldap</Val>
    <Val>cHTML|Ldap</Val>
   <Val>Nokia|Unix;SecureID</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

In the above example, the mobile client Nokia users are setup to use only UNIX and SecureID for authentication. Hence, when a user authenticates using a Nokia, the user is presented with an authentication menu that allows the user to log in using UNIX or SecureID.

By default, LDAP is the only supported authentication module for mobile devices.

NOTE The login worker class can be specific to a clientType and can serve content for different clientTypes. For example, if the clientType is WML, there can be a `WmlLoginWorker` class that can retrieve content specific to WML clients. See `iwtAuth-loginWorkerClass` and `iwtAuth-loginWorkerClasses` for more information.

iwtAuth-loginWorkerClass

The following is an example of the `iwtAuth-LoginWorkerClass` attribute. This attribute stores the loginWorkerClassName for HTML clients only. By default, `com.iplanet.portalserver.auth.server.WirelessLoginWorker` is the value for this attribute.

```
<iwt:Att name="iwtAuth-loginWorkerClass"
desc="Client Specific Auth Page Generator Classes"
type="stringlist"
idx="X-x27"
userConfigurable="FALSE">
<Val>default|com.iplanet.portalserver.auth.server.WirelessLoginWork
er</Val>
<Rperm>ADMIN</Rperm>
<Wperm>ADMIN</Wperm>
</iwt:Att>
```

iwtAuth-loginWorkerClasses

The following is an example of the `iwtAuth-loginWorkerClasses` attribute. This attribute identifies which clientType must use which loginWorker class. This attribute is a stringlist storing the clientType and the loginWorker class name (`clientType|loginWorkerClassName`) for the specified client. By default, no values are defined in this attribute and the loginWorker class name is derived from `iwtAuth-loginWorkerClass` attribute.

```
<iwt:Att name="iwtAuth-loginWorkerClasses"
type="stringlist"
desc="Pluggable Auth page generator classes for clients"
idx="X-x31"
userConfigurable="false">
```

```
<Val>genericHTML|com.iplanet.portalserver.auth.server.HTMLLoginWork
er</Val>

<Val>WML|com.iplanet.portalserver.auth.server.WirelessLoginWorker</
Val>

<Val>cHTML|com.iplanet.portalserver.auth.server.WirelessLoginWorker
</Val>

<Wperm>ADMIN</Wperm>

<Rperm>ADMIN</Rperm>

</iwt:Att>
```

By default, this attribute uses the `com.iplanet.portalserver.auth.server.WirelessLoginWorker` class as the attribute value. The `WirelessLoginWorker` class also supports WML and cHTML clients.

Client Detection Module Attributes

The attributes discussed in Table 8-11 reside in the profile component `iwtClient`.

Table 8-11 iwtClient Attributes

Attribute	Type	Description	Acceptable Value
`iwtClient-defaultClientType`	string	This attributes stores the clientType supported by default.	*clientType*
`iwtClient-clientTypes`	stringlist	This attribute stores some critical client specific information for retrieving content specific to the client.	*name=value*

iwtClient-defaultClientType

The following is an example of the `iwtClient-defaultClientType` attribute. This attribute stores the clientType that is supported by default. The default clientType supported by the Mobile Access Pack server is generic HTML.

```
<iwt:Att name="iwtClient-defaultClientType"

    type="string"

    desc="default Client Type"

    idx="a101"
    userConfigurable="false">
    <Val>genericHTML</Val>
    <Wperm>ADMIN</Wperm>
    <Rperm>ADMIN</Rperm>
</iwt:Att>
```

iwtClient-clientTypes

The following is an example of the `iwtClient-clientTypes` attribute.

```
<iwt:Att name="iwtClient-clientTypes"
type="stringlist"
desc="Client Types"
idx="a100"
userConfigurable="false">
<Val>clientType=genericHTML|userAgent=NeverWillMatchAUserAgent|cont
entType=text/html|fileIdentifier=html|filePath=html|genericHTML=tru
e</Val>
<Val>clientType=WML|userAgent=UP|contentType=text/vnd.wap.wml|fileI
dentifier=wml|filePath=wml/UP|counterName=C|escapeWML=true|lineLeng
th=20|charsets=ISO-8859-1;UTF-8</Val>
<Val>clientType=cHTML|userAgent=DoCo|contentType=text/html|fileIden
tifier=chtml|filePath=chtml|escapeXML=true|lineLength=20|charsets=S
hift_JIS</Val>
<Val>clientType=pana_p503i|contentType=text/html|fileIdentifier=cht
ml|filePath=chtml/pana_p503i|genericHTML=false|userAgent=P503i|esca
peXML=true|lineLength=20|charsets=Shift_JIS</Val>
<Val>clientType=siemens_s35|contentType=text/vnd.wap.wml|fileIdenti
fier=wml|filePath=wml/UP|genericHTML=false|userAgent=SIE-S35|counte
rName=C|escapeWML=true|lineLength=20|charsets=UTF-8;ISO-8859-1</Val
>
<Val>clientType=nokia_7110|contentType=text/vnd.wap.wml|fileIdentif
ier=wml|filePath=wml/Nokia/Nokia_7110|genericHTML=false|userAgent=N
okia7110|counterName=C|escapeWML=true|lineLength=20|maxSize=2500|ch
arsets=ISO-8859-1</Val>
<Wperm>ADMIN</Wperm>
<Rperm>ADMIN</Rperm>
</iwt:Att>
```

This attribute is a string list with an embedded list of name=value pairs within each list element. Each list element contains one or more or the following information about the client:

- **charset** - The charset element contains a supported (semicolon delimited list of) character sets for the corresponding client. The list specified here is displayed in the Edit Options page (on the user's desktop) for the corresponding client. This property's value must be specified when configuring the client.
- **clientType** - The client type can be stored in this attribute as an arbitrary string (with a list of name=value pairs) uniquely identifying the client. This attribute serves as an index to the rest of the Client Data. By default, the following clients are supported:

clientType	Supported Clients
WML	Openwave browsers using WML
cHTML	DoCoMo phones using cHTML
nokia_7110	Nokia 7110 phone using WML
pana_p503i	Panasonic P503i phones using cHTML
siemens_s35	Siemens S35 phone using WML

- **contentType** - The contentType element contains the HTTP header value (such as text, html, wml). It specifies the acceptable content type for the requesting client. The contentType header is retrieved from the profile service and it can specified as a forward slash delimited list of acceptable content types. For fetching the contentType, the default clientType is used if client detection is disabled. This property's value must be specified when configuring the client.
- **counterName** - This is used to add a counter to the URL.
- **escapeWML** - This attribute is used by certain providers to handle content that contains special characters. This is a boolean attribute. If set to true, the special characters discussed below are escaped. The special characters that will be escaped within the provider content are:

Description	Symbol	Named Entity
Dollar	$	$$
Apostrophe	'	&apos;

Description	Symbol	Named Entity
Quote	"	&quot;
Ampersand	&	&amp;
Less Than	<	&lt;
Greater Than	>	&gt;

- **escapeXML** : This attribute is used by certain providers to handle content that contains special characters. This is a boolean attribute. If set to true, the special characters listed below are escaped. The special characters that will be escaped within the provider content are:

Description	Symbol	Named Entity
Quote	"	&quot;
Ampersand	&	&amp;
Less Than	<	&lt;
Greater Than	>	&gt;

- **fileIdentifier** - The fileIdentifier element indicates the supported file types for the specified client.
- **filePath** - The filePath element indicates the path to the client specific data files (such as templates and JSP files). Template and JSP files reside in sub-directories based on the desktop type, locale, and component (see "File Lookup Mechanism," on page 18 for more information). For client specific file lookup, the search is based on the client's filePath attribute specified here. This property's value must be specified when configuring the client.
- **genericHTML** - This attribute specifies whether or not the client accepts standard HTML.
- **ignoreHostHeader** - This attribute makes the desktop ignore the host header. This is particularly useful when the host sends a HTTP/1.0 header.
- **lineLength** -This specifies the number of characters that may appear on a single line of the device's display area.

- **maxSize** - This is used to constrain the output size of the WirelessFrontProvider. It specifies the approximate number of bytes to send to the mobile device. The WirelessFrontProvider filters out the content restricting it to the maxSize limit specified here.
- **userAgent** - The userAgent attribute stores a search string or filter. It is used to compare or match against the client's HTTP user agent header to determine the clientType. That is, the client detection module iterates over the supported list of clientTypes, comparing their userAgent against the user agent header of the client accessing the server. This property's value must be specified when configuring the client.

Desktop Attributes

The attributes discussed in reside in the profile component `iwtDesktop`.

Table 8-12 iwtDesktop Attributes

Attribute	Type	Description	Acceptable Value
`iwtDesktop-clientAllProviders`	stringlist	This attribute indicates the list of providers supported for the specified client.	clientType|channel1;channel2
`iwtDesktop-clientUsersSelectedProviders`	stringlist	This attribute indicates the list of providers from the list of supported providers that are made available to the client.	clientType|channel1;channel2
`iwtDesktop-clientChannelListModules`	stringlist	This attribute stores the channel list module for each supported client.	clientType|com.iplanet.portalserver.desktop.util.channellist.ClientChannelList

iwtDesktop-clientAllProviders

The following is an example of `iwtDesktop-clientAllProviders` attribute. This attribute stores a list of available channels for each supported client. This attribute is used to determine which channels are available to a given clientType.

```
<iwt:Att name="iwtDesktop-clientAllProviders"

type="stringlist"
```

```
desc="Client Specific Available Providers"

idx="X-x109"

userConfigurable="true">

<Val>nokia_7110|iwtUserInfoProvider;iwtXMLProvider;iwtWirelessBookm
arkProviderWML;iwtWirelessPersonalNoteProvider;iwtPostitProvider;iw
tCalendarProvider;iwtMailProvider;iwtAddressBookProvider</Val>

<Val>siemens_s35|iwtUserInfoProvider;iwtXMLProvider;iwtWirelessBook
markProviderWML;iwtWirelessPersonalNoteProvider;iwtPostitProvider;i
wtCalendarProvider;iwtMailProvider;iwtAddressBookProvider</Val>

<Val>pana_p503i|iwtUserInfoProvider;iwtXMLProvider;iwtWirelessBookm
arkProvidercHTML;iwtWirelessPersonalNoteProvider;iwtPostitProvider;
iwtCalendarProvider;iwtMailProvider;iwtAddressBookProvider</Val>

<Val>WML|iwtUserInfoProvider;iwtXMLProvider;iwtWirelessBookmarkProv
iderWML;iwtWirelessPersonalNoteProvider;iwtPostitProvider;iwtCalend
arProvider;iwtMailProvider;iwtAddressBookProvider</Val>

<Val>cHTML|iwtUserInfoProvider;iwtXMLProvider;iwtWirelessBookmarkPr
ovidercHTML;iwtWirelessPersonalNoteProvider;iwtPostitProvider;iwtCa
lendarProvider;iwtMailProvider;iwtAddressBookProvider</Val>

<Wperm>ADMIN</Wperm>

<Rperm>ADMIN</Rperm>

<Rperm>OWNER</Rperm>

</iwt:Att>
```

iwtDesktop-clientUserSelectedProviders

The following is an example of the `iwtDesktop-clientUserSelectedProviders` attribute. This attribute stores a list of selected channels for each supported client. This attribute is used to determine which channels the user has selected to show up on a given client. If no channel is selected, the value of this attribute corresponding to the given clientType is removed from the profile.

```
<iwt:Att name="iwtDesktop-clientUserSelectedProviders"

    type="stringlist"

    desc="Client Specific Selected Providers"

    idx="X-x110"

    userConfigurable="true">

    <Val>nokia_7110|iwtUserInfoProvider</Val>

    <Val>siemens_s35|iwtUserInfoProvider</Val>
    <Val>pana_p503i|iwtUserInfoProvider</Val>

    <Val>WML|iwtUserInfoProvider</Val>

    <Val>cHTML|iwtUserInfoProvider</Val>

    <Wperm>ADMIN</Wperm>

    <Rperm>ADMIN</Rperm>

</iwt:Att>
```

iwtDesktop-clientChannelListModules

The following is an example of the `iwtDesktop-clientChannelListModules` attribute. The iPlanet Portal Server channel list module provides a mechanism to store and retrieve multiple lists of selected and available channels.

The client channel list module extends the support for separate available and selected lists for each supported clientType. This allows users to specify what channels to display on their devices.

This attribute stores the client specific channel list modules to enable selection of channel by each user for each one of the devices setup for them.

```
<iwt:Att name="iwtDesktop-clientChannelListModules"

type="stringlist"

desc="Client Specific Channel List Modules"

idx="X-x111"

userConfigurable="true">

<Val>nokia_7110|com.iplanet.portalserver.desktop.util.channellist.C
lientChannelList</Val>

<Val>siemens_s35|com.iplanet.portalserver.desktop.util.channellist.
ClientChannelList</Val>

<Val>pana_p503i|com.iplanet.portalserver.desktop.util.channellist.C
lientChannelList</Val>

<Val>WML|com.iplanet.portalserver.desktop.util.channellist.ClientCh
annelList</Val>

<Val>cHTML|com.iplanet.portalserver.desktop.util.channellist.Client
ChannelList</Val>

<Wperm>ADMIN</Wperm>

<Rperm>ADMIN</Rperm>

</iwt:Att>
```